INAUGURAL – DISSERTATION

zur Erlangung der Doktorwürde
der
Naturwissenschaftlich–Mathematischen Gesamtfakultät
der
Ruprecht – Karls – Universität
Heidelberg

vorgelegt von

Dipl.–Math. Peter Riede aus Mannheim

Tag der mündlichen Prüfung: 12.06.2006

Optimierung von dynamischen Multiple-Setpoint-Problemen mit Anwendung bei Fahrzeugmodellen

Gutachter: Prof. Dr. Dr. h.c. Hans Georg Bock

Gutachter: Prof. Dr. Gerhard Reinelt

Zusammenfassung

Die vorliegende Arbeit präsentiert eine neue anwendungsgerechte Methode für nichtlineare Optimierungsprobleme. Die Formulierung eines sogenannten Multiple-Setpoint-Problems führt zu Lösungen, die in der praktischen Anwendung oft deutlich bessere Ergebnisse erzielen als bei der üblichen Formulierung eines Optimierungproblems, da für einen ganzen Bereich (Multiple-Setpoint-Lösung) optimiert wird und nicht für einen einzigen festen Parametersatz (Single-Setpoint-Lösung). Gesucht sind Lösungen, die in dem Sinne optimal für verschiedene Szenarien oder für einen ganzen Parameterbereich sind, daß sie Nebenbedingungen für alle Szenarien einhalten und eine Zielfunktion in einem Mittel oder in einem Worst Case minimieren. Obwohl solche Lösungen in vielen Bereichen von großem Nutzen sind, ist die Multiple-Setpoint-Optimierung ein bisher wenig untersuchter Bereich.

Zur Untersuchung von Optimierungsstrategien wird eine Problemklasse herausgegriffen – die Probleme der optimalen Steuerung von Systemen, die sich durch differentiell-algebraische Gleichungen (DAE) beschreiben lassen. Mithilfe des Randwertproblemansatzes und einer Diskretisierung durch Bocks direkte Mehrziel-Methode (Bock's direct multiple shooting) werden die Probleme in große endlich-dimensionale nichtlineare Optimierungsprobleme transformiert, die mit einem SQP-Verfahren gelöst werden können.

In dieser Arbeit wird eine allgemeine mathematische Problemformulierung für die Klasse der Multiple-Setpoint-Probleme und ein neuartiger, strukturausnutzender SQP-Algorithmus für Multiple-Setpoint-Probleme aus dem Bereich der optimalen Steuerung gegeben. Der Algorithmus ermöglicht es, neue Problemklassen zu behandeln bzw. Lösungen zu produzieren, die für den praktischen Einsatz in vielen Fällen geeigneter sind als bei bisherigen Optimierungsansätzen.

Der Einsatz des neuen Algorithmus wird an einem komplexen Problem aus der Industrie demonstriert. In einem Auto soll durch eine Optimierung der Motorlagerung die Schwingungsübertragung von Straße und Motor auf den Fahrersitz in einem Frequenzbereich minimiert und somit der Fahrkomfort erhöht werden. Mithilfe einer (3-dimensionalen) Mehrkörpersimulation in natürlichen Koordinaten wird ein Automodell mit 32 kinematischen Freiheitsgraden betrachtet. Eine Fourier-Analyse hilft, die Schwingungsübertragung zu untersuchen. Das entstehende nichtlineare Optimierungsproblem hat eine Dimension von mehreren tausend Variablen. Durch Anwendung des neuen Algorithmus kann (unter Einhaltung von gewissen Nebenbedingungen) eine optimierte Motorlagerung gefunden werden, die eine Reduktion der Schwingungsübertragung im vorgegebenen Frequenzbereich um ca. 80% gegenüber der gegebenen Startlösung erzielt.

Die Implementierung des neuen Multiple-Setpoint-Algorithmus' wird in einem neuen Softwarepaket realisiert, das bei der Firma Freudenberg im industriellen Einsatz ist. Die entwickelte Software basiert auf der Optimierungssoftware für Probleme der optimalen Steuerung MUSCOD-II sowie dem objekt-orientierten Modellierungstool für Mehrkörpersysteme MBSNAT, mit dem die Modellierung des Fahrzeugs inklusive der Berechung der für die Optimierung nötigen Ableitungen geschieht.

Abstract

This thesis presents a new method for nonlinear optimization problems which meets the requirements of industrial applications. The formulation of a so-called Multiple Setpoint Problem leads to solutions that produce significantly better results in practice than a conventional formulation of an optimization problem because it is optimized for a whole range (multiple setpoint solution) instead of a single fixed set of parameters (single setpoint solution). Solutions of this formulation are either optimal for different scenarios or for a whole range of parameters in the sense that constraints are satisfied for all scenarios and an objective function is minimized on an average or in the worst case. Although such solutions are of use in many areas, the multiple setpoint optimization is a field seldom explored so far.

For the examination of optimization strategies one problem class is given special attention – the optimal control problems for systems described by differential-algebraic equations (DAE). Using the boundary value problem approach and a discretization by Bock's direct multiple shooting method the problems are transformed into large-scale finite-dimensional nonlinear optimization problems that can be solved by an SQP-algorithm.

In this thesis a general mathematical problem formulation for the class of multiple setpoint problems and a new, structure exploiting SQP-algorithm for multiple setpoints problems in optimal control is given. The algorithm allows for treating new problem classes and producing solutions that are in many cases more suitable for practical use than in the existing approaches.

The application of the new algorithm is demonstrated on a complex industry problem. In a car the vibration transmission from the street and the engine to the driver's seat is to be minimized in a wide frequency range by optimizing the mounting of the engine in order to increase driving comfort. With a (3-dimensional) multibody simulation in natural coordinates a car model with 32 kinematical degrees of freedom is studied. A Fourier analysis helps to examine the vibration transmission. The resulting optimization problem has a dimension of several thousand variables. By use of the new algorithm an optimization of the mounting of the engine (in compliance with some restrictions) can be found that leads to a reduction of vibration transmission in the specified frequency range of approximately 80% compared to the given reference solution.

The implementation of the new multiple setpoint algorithm was realized in a new software package that is in industrial use at the company Freudenberg. The developed software is based on the optimization tool for optimal control problems MUSCOD-II and the object-orientated modeling tool for multibody systems MBSNAT, which is used for the modeling of the car including the calculation of the derivatives necessary for the optmization.

Danksagung

Diese Arbeit wurde am Interdiszipliären Zentrum für Wissenschaftliches Rechnen (IWR) an der Universität Heidelberg in der Arbeitsgruppe “Simulation und Optimierung” von Prof. Dr. Dr. h.c. Hans Georg Bock erstellt. Ich möchte mich an dieser Stelle bedanken bei meinen Betreuern Prof. Dr. Dr. h.c. Hans Georg Bock und Dr. Johannes P. Schlöder für die ausgezeichnete Unterstützung in den vergangenen Jahren, für anregende Diskussionen und zahlreiche hilfreiche Ratschläge. In meiner Zeit am IWR durfte ich sowohl von dem exzellenten Fachwissen in der Arbeitsgruppe im Bereich der Simulation und Optimierung von dynamischen Prozessen als auch von der kollegialen und produktiven Arbeitsatmosphäre profitieren. Insbesondere stellte die in der Arbeitsgruppe entwickelte Optimal-Steuerungs-Software MUSCOD-II die Grundlage für die Implementierung der neuen Methoden dar.
Ebenso möchte ich all meinen Kollegen in der Arbeitsgruppe danken für die freundschaftliche Zusammenarbeit. Mehrere Kollegen haben besonders zu dem Erfolg dieser Arbeit beigetragen. Als erstes danke ich meinem Bürokollegen Sebastian Sager für die gemeinsame Zeit in Zimmer 405 und besonders für das Korrekturlesen der kompletten Arbeit. Christian Kraus danke ich für die Beantwortung zahlreicher Fragen bei der Einarbeitung in die Mehrkörper-Software MBSNAT. Moritz Diehl danke ich für die Mentorschaft und für die kreativen Ideen bei wissenschaftlichen Problemen. Andreas Schäfer danke ich für die Beantwortung unzähliger Fragen beim Umgang mit der Optimierungs-Software MUSCOD-II. Stefan Körkel danke ich für ein zu jeder Zeit offenes Ohr und für die Unnachgiebigkeit, mit der er zu helfen wußte. Außerdem danke ich recht herzlich allen Korrekturlesern.
Weiterhin danke ich dem Unternehmen “Freudenberg Dichtungs- und Schwingungstechnik GmbH und Co. KG” (Weinheim), insbesondere Dipl.-Phys. Hansjörg Maier und Dr. Jörg Hechenblaikner für die Bereitstellung des Anwendungsbeispiels und für die fruchtbare Zusammenarbeit innerhalb des Industrieprojektes.
Mein persönlicher Dank gilt meinen Eltern Dr. Adolf und Karin Riede für ihre fortwährende Unterstützung.

Inhaltsverzeichnis

Abbildungsverzeichnis

Notation

Mathematische Symbole

Hier sind die wichtigsten mathematischen Symbole aufgeführt.

Allgemeines

$\nabla f := \left(\frac{\partial}{\partial x_1} f(x), \ldots, \frac{\partial}{\partial x_n} f(x)\right)$	Gradient einer Funktion $f : \mathbb{R}^n \to \mathbb{R}$
$\nabla^2 f := \left(\frac{\partial^2}{\partial x_i x_j} f(x)\right)_{i,j=1,\ldots,n}$	Hesse-Matrix einer Funktion $f : \mathbb{R}^n \to \mathbb{R}$
$f_x(x,y) := \frac{\partial}{\partial x} f(x,y)$	partielle Ableitung einer Funktion
$\|x\|$	euklidische Norm
I	Einheitsmatrix

Probleme der optimalen Steuerung

$x(t) \in \mathbb{R}^{n_x}$	differentielle Variable bei DAEs
$z(t) \in \mathbb{R}^{n_z}$	algebraische Variable bei DAEs
$u(t) \in \mathbb{R}^{n_u}$	Steuerfunktion
p, p_{ij}	Parameter
t	Zeit
t_0	Startzeitpunkt
t_f	Endzeitpunkt
t_k	Zeitpunkt im Zeitgitter
h_k	Schrittweite
σ	lokaler Fehler
τ	Konsistenzfehler
o	Konsistenzordnung bzw. Konvergenzordnung
ε	globaler Fehler
η_k	berechneter Wert der Funktion y an einem Gitterpunkt

$y(t)$	zu integrierende Funktion
Φ	Mayer-Term der Zielfunktion
ϕ	Lagrange-Term der Zielfunktion
M	Anzahl der Stufen
d_i	Stufendauern
m_i	Anzahl Mehrzielknoten auf Stufe i
s_{ij}^x	Parametrisierung der differentiellen Zustände
s_{ij}^z	Parametrisierung der algebraischen Zustände
χ_{ij}	Basisfunktionen zur Parametrisierung der Steuerung
q_{ij}	Parameter der diskretisierten Steuerfunktionen
p_{ij}	lokalisierte Parameter
τ_{ij}	dimensionsloses Zeitgitter
v	Zusammenfassung von t_0 und den Stufendauern d_i
$\theta_i(\tau,v)$	Zeittransformation
I_{ij}	Mehrzielintervall
$\hat{q} := (q_{ij}, p_{ij}, v_{ij})$	Zusammenfassung der Steuergrößen
c_i	Stufenübergangsbedingungen
r^s,r^e, r_{ij}	Randwert- bzw. Mehrpunktbedingungen
h	Konsistenzbedingungen

Nichtlineare Probleme

$D \subseteq \mathbb{R}^n$	Gebiet in $\mathbb{R}^n$, Definitionsmenge für F,G,H
$F : D \subseteq \mathbb{R}^n \to \mathbb{R}$	Zielfunktion im allgemeinen NLP
$G : D \subseteq \mathbb{R}^n \to \mathbb{R}^l$	Gleichungsbeschränkung im allgemeinen NLP
$H : D \subseteq \mathbb{R}^n \to \mathbb{R}^m$	Ungleichungsbeschränkung im allgemeinen NLP
$S \in \mathbb{R}^n$	zulässige Menge
$w^* \in \mathbb{R}^n$	Lösung
$\mathcal{L}$	Lagrange-Funktion
λ, μ	Lagrange-Multiplikatoren
k	Iterationsindex
$F_k, G_k, H_k, \mathcal{L}_k$	$F(w_k), G(w_k), H(w_k), \mathcal{L}(w_k, \lambda_k, \mu_k)$
B_k	Approximation der Hesse-Matrix in Iteration k
T	Gütefunktion

Grundlegende Definitionen

DEFINITION 1

1. Eine Matrix $A \in \mathbb{R}^{n \times n}$ heißt positiv definit, wenn gilt:

$$x^T A x \;>\; 0 \quad \forall\, x \in \mathbb{R}^n,\; x \neq 0$$

2. Eine Matrix $A \in \mathbb{R}^{n \times n}$ heißt positiv semidefinit, wenn gilt:

$$x^T A x \;\geq\; 0 \quad \forall\, x \in \mathbb{R}^n$$

DEFINITION 2

1. (x^k) konvergiert linear, wenn gilt:
   $\exists c < 1 : \quad \|x^{k+1} - x^*\| \;\leq\; c\|x^k - x^*\| \quad \forall\, k \geq k_0$

2. (x^k) konvergiert superlinear, wenn gilt:
   $\|x^{k+1} - x^*\| \;\leq\; c_k\|x^k - x^*\| \quad \forall\, k \geq k_0 \quad$ und $\quad c_k \longrightarrow 0$

3. (x^k) konvergiert quadratisch, wenn gilt:
   $\exists D : \quad \|x^{k+1} - x^*\| \;\leq\; D\|x^k - x^*\|^2 \quad \forall\, k \geq k_0$
   (”Anzahl gültiger Stellen verdoppelt sich von Schritt zu Schritt”)

4. (x^k) konvergiert von der Ordnung p ($p > 1$), wenn gilt:
   $\exists D : \quad \|x^{k+1} - x^*\| \;\leq\; D\|x^k - x^*\|^p \quad \forall\, k \geq k_0$

Abkürzungen

ODE	Ordinary Differential Equation, gewöhnliche Differentialgleichung
DAE	Differential-Algebraic Equation, differentiell-algebraische Gleichung
NLP	Nichtlineares Programm
KKT-Punkt	Karush-Kuhn-Tucker-Punkt
QP	Quadratisches Programm, Quadratisches Problem
SQP	Sequential Quadratic Programming, sequentielle quadratische Programmierung
BFGS-Update	Updateformel von Broyden, Fletcher, Goldfarb und Shanno
DFP-Update	Updateformel von Davidon, Fletcher und Powell

0 Einleitung

In den letzten Jahren fand ein bemerkenswerter Fortschritt im Bereich der Modellierung, Simulation und Optimierung von dynamischen Prozessen statt. Neue numerische Methoden, zunehmend leistungsfähigere Computer und interdisziplinäre Forschungsprojekte machen es möglich, daß immer komplexere Prozesse aus Physik, Chemie, Biologie, Medizin, Ingenieurswesen und weiteren Disziplinen modelliert und analysiert werden können. Oft führt dies zu Modellen in Form von großen, steifen und nichtlinearen Systemen von differentiell-algebraischen Gleichungen (DAE). Die industrielle Einsetzbarkeit der Verfahren und die Aussicht, durch mathematische Optimierung effizienter produzieren, qualitativ verbesserte Produkte anbieten oder Energie bzw. Geld einsparen zu können, machen die Optimierung dynamischer Systeme heute zu einem der wichtigsten Forschungsgebiete.
Aufgrund des starken Wettbewerbs auf dem globalisierten Markt wird es zunehmend wichtiger, nicht nur schnelle Algorithmen für immer größere Probleme bereitzustellen, sondern auch zu garantieren, daß die gewonnenen Lösungen an die in der Industrie gegebenen Voraussetzungen angepaßt sind. Ein wichtiger Punkt ist es, robuste Lösungen zu erlangen, die nicht nur im simulierten Testfall gut funktionieren, sondern auch bei den in der Praxis oft wechselnden Rahmenbedingungen gute Ergebnisse erzielen.
Robuste Optimierung beschäftigt sich damit, Lösungen zu finden, die auch für leicht gestörte Parameter zulässig sind, und entwirft mit Ansätzen der Wahrscheinlichkeitsanalyse Strukturen, die bei begrenzter Variation der Entwurfsparameter keine Einschränkung der Funktionalität beherbergen. Die robuste Optimierung soll Lösungen liefern, die keine oder nur eine sehr geringe Sensivität gegenüber Unsicherheiten bzw. Variationen in den Modelldaten aufweisen. Eine Einführung in robuste bzw. stochastische Optimierung bei nichtlinearen Problemen liefern [SR03], [Zha05] oder [BTN02]. Zu robuster Optimierung bei Problemen der optimalen Steuerung verweisen wir auf [KKBS04], [Die04] und [DBK06].
In dieser Arbeit wollen wir einen Schritt weiter gehen und einen Algorithmus vorstellen, der von vornherein eine Optimierung für verschiedene Szenarien oder eine Optimierung in einem ganzen Parameterbereich vornimmt. Mit der mathematischen Formulierung eines sogenannten “Multiple-Setpoint-Problems” lassen sich zwei Fälle abdecken: erstens das optimale Design für Systeme mit unterschiedlichen Einsatzmöglichkeiten, wie zum Beispiel das Design eines Roboters mit verschiedenen Aufgaben; und zweitens die Optimierung von Systemen, bei denen ein Prozeß unter schwankenden Rahmenbedingungen abläuft, das heißt, daß bestimmte Parameter Werte aus einem vorgegebenem Bereich annehmen können. Die simultane Optimierung der Setpoints garantiert nicht nur, daß eine “globale Minimierung” der Zielfunktion für alle gegebenen Fälle erreicht wird, sondern auch, daß überhaupt Zulässigkeit der Lösung in allen Fällen gewährleistet ist.

Im Gegensatz zur “Mehrzieloptimierung” ([Mey04]) wird hier nicht ein Prozeß in Hinblick auf mehrere Ziele optimiert, die sich zum Teil widersprechen (z.B. minimaler Kraftstoffverbrauch und maximale Beschleunigung bei Autos), sondern mehrere Prozesse (bzw. der gleiche Prozeß bei wechselnden Rahmenbedingungen) gleichzeitig optimiert.
In dieser Arbeit wird ein Bogen gespannt von einem Abriß der Probleme der optimalen Steuerung und deren Lösung mit speziell zugeschnittenen SQP-Methoden über eine Formulierung der mathematischen Problemstellung von Multiple-Setpoint-Problemen und Bereitstellung eines neuen numerischen SQP-Algorithmus für Multiple-Setpoint-Probleme und deren Implementierung in einem für den praktischen Einsatz geeigneten Softwarepaket bis zu numerischen Ergebnissen im praktischen Einsatz des Algorithmus anhand eines anspruchsvollen Beispiels aus der Automobilindustrie.
Im Bereich der Anwendung wurde ein Problem behandelt, das bisher noch nicht gelöst werden konnte. Während es für die Übertragung von hochfrequenten Schwingungen (im menschlichen Hörbereich) in einem Fahrzeug schon Veröffentlichungen gibt, die auf statistischer Energie-Analyse beruhen ([PCC97]), ist die Optimierung von Übertragung von Schwingungen im untersuchten Bereich $2 - 25$ Hz ein kaum behandeltes Feld. Auch wurden bei den bisherigen Untersuchungen nur Simulation und Analyse durchgeführt, nicht aber Optimierung.

Die Arbeit ist wie folgt in fünf Kapitel gegliedert:

Das erste Kapitel führt in die Probleme der optimalen Steuerung ein. Nach einer Behandlung von differentiell-algebraischen Gleichungen und der Lösung von Anfangswertproblemen durch lineare Mehrschritt-Methoden wird eine allgemeine mathematische Formulierung für ein Problem der optimalen Steuerung gegeben. Auf eine Diskussion verschiedener Lösungsansätze folgt der zentrale Punkt des Kapitels, die Vorstellung von Bocks direkter Mehrziel-Methode (Bock’s direct multiple shooting method).
Im zweiten Kapitel beschäftigen wir uns mit Lösungsmethoden für die endlich-dimensionalen, beschränkten Optimierungsprobleme, die aus der Mehrziel-Methode resultieren, den SQP-Verfahren. Es werden die Optimalitätsbedingungen formuliert, auf denen die SQP-Verfahren arbeiten und die Details des SQP-Algorithmus, speziell auch neuere Ansätze, beleuchtet. Abschließend wird ein speziell auf die Struktur der Probleme der optimalen Steuerung zugeschnittener SQP-Algorithmus formuliert, der im Softwarepaket MUSCOD-II implementiert ist.
Das dritte Kapitel ist der Einführung der Problemklasse von Multiple-Setpoint-Problemen gewidmet. Ausgehend von einer Motivation, wo Multiple-Setpoint-Probleme auftreten und warum diese Problemformulierung in vielen Fällen zu Lösungen führt, die für den Einsatz in der Praxis große Vorteile bieten, wird zunächst eine allgemeine mathematische Problemformulierung gegeben, die dann spezialisiert wird für den Bereich der Probleme der optimalen Steuerung. Für diese Problemklasse wird ein neuer, strukturausnutzender und effizienter Algorithmus gegeben, der es möglich macht, neue Problemklassen zu behandeln bzw. Lösungen zu produzieren, die für den praktischen Einsatz in vielen Fällen geeigneter sind als bei bisherigen Optimierungsansätzen.
Im vierten Kapitel wird der Einsatz dieses Algorithmus’ an einem komplexen Problem aus der Automobilindustrie demonstriert, das bisher noch nicht gelöst werden konnte. Mithilfe einer Mehrkörpersimulation in natürlichen Koordinaten wird ein Automodell mit 32 kinematischen

Freiheitsgraden betrachtet. Eine Fourier-Analyse hilft, die Schwingungsübertragung von Straße und Motor auf den Fahrersitz in einem großen Frequenzbereich zu untersuchen. Das entstehende nichtlineare Optimierungsproblem hat eine Dimension von mehreren tausend Variablen. Durch Anwendung des neuen Algorithmus konnte eine Optimierung der Motorlagerung gefunden werden, die die Schwingungsübertragung im vorgegebenen Frequenzbereich gegenüber der gegebenen Startlösung um ca. 80% reduziert. Die entwickelte Software ist bei der Firma Freudenberg im praktischen Einsatz.

1 Probleme der optimalen Steuerung

In diesem Kapitel stellen wir die *Probleme der optimalen Steuerung (optimal control problems)* vor, die als Problemklasse herausgegriffen wurden, um das Multiple-Setpoint-Konzept anzuwenden. Grundlage der Probleme der optimalen Steuerung sind Modelle, deren Dynamik sich durch *differentiell-algebraische Gleichungen (differential algebraic equation, DAE)* beschreiben läßt.
Diese Problemklasse deckt einen weiten Bereich der Probleme ab, die in der Industrie auftauchen. Anwendungen finden sich zum Beispiel in den Bereichen Ingenieurswesen ([BEKS02],), Chemie ([BP03], [Mar02], [KKEvS01], [KvSB01], [Lei99], [BBLS99], [RBP+99], [Die98],[Diw95]), Biotechnologie ([BP03]) und Finanzmathematik ([WBPMS04]).
Es können auch besonders schwere Probleme gelöst werden, zum Beispiel dynamische Optimierung mit Unstetigkeiten ([BP04]), NMPC (nonlinear model predictive control) bzw. Echtzeitoptimierung ([DFS+02], [Die02], [BDLS00]), robuste Optimierung ([KKBS04]), gemischtganzzahlige optimale Steuerung ([Sag05]), oder Optimierung bei parabolischen Randwertproblemen, bei der die partielle Differentialgleichung unter Anwendung der Linien-Methode (method of lines) in eine DAE übergeführt wird ([Sch04]).
Das Multiple-Setpoint-Konzept ließe sich aber auch auf beliebige andere nichtlineare Probleme anwenden, zum Beispiel auf Modelle, die auf partiellen Differentialgleichungen (PDE) oder Funktionaldifferentialgleichungen beruhen, oder auch für ganz allgemeine nicht-zeitabhängige Probleme.
Im ersten Abschnitt werden Eigenschaften und numerische Methoden, speziell lineare Mehrschritt-Verfahren, zur Lösung von Anfangswertproblemen bei differentiell-algebraischen Gleichungen vorgestellt.
Im zweiten Abschnitt wird eine allgemeine mathematische Formulierung für ein Problem der optimalen Steuerung gegeben sowie die verschiedenen gebräuchlichen Lösungsansätze aufgezeigt.
Im dritten Abschnitt wird *Bocks direkte Mehrziel-Methode* (Bock's direct multiple shooting) im Detail vorgestellt, mithilfe derer das kontinuierliche Problem in verschiedenen Parametrisierungs- und Diskretisierungsschritten durch ein strukturiertes endlich-dimensionales nichtlineares Problem (nonlinear program, NLP) ersetzt wird.
Eine detailliertere Diskussion von Problemen der optimalen Steuerung findet sich zum Beispiel in [Lei99] und [Sag05].

1.1 Differentiell-algebraische Gleichungen

Dynamische Prozesse aus den Naturwissenschaften und aus der Industrie lassen sich oft durch ein System gewöhnlicher Differentialgleichungen beschreiben. Häufig sind dabei die Zustands-

variablen Nebenbedingungen unterworfen, zum Beispiel durch Erhaltungsgesetze, geometrische Einschränkungen oder Invarianten. Dadurch können die Trajektorien nicht den gesamten Zustandsraum durchlaufen, sondern bewegen sich auf Mannigfaltigkeiten. Eine Möglichkeit ist es, durch geschickte Wahl der Variablen die Nebenbedingungen zu eliminieren und nur die *gewöhnlichen Differentialgleichungen (ordinary differential equation, ODE)* zu betrachten. Dabei können stark nichtlineare Systeme entstehen, die schwer zu lösen sind. Ein anderer Ansatz behandelt die algebraischen Nebenbedingungen als Teil des dynamischen Systems. Man nennt diese Systeme *differentiell-algebraische Gleichungen (differential algebraic equation, DAE)*.
Tiefergehendere Informationen zu differentiell-algebraischen Gleichungen findet man in zahlreichen Lehrbüchern, zum Beispiel [Asc98], [HW93] und [BCP96], oder auch in [Kör02] und [BP04].

1.1.1 Klassifizierung von differentiell-algebraischen Gleichungen

Die allgemeinste Formulierung einer differentiell-algebraischen Gleichung ist die *voll-implizite nichtlineare DAE*:

$$F(t,x,\dot{x}) \;=\; 0 \quad t \in [t_0,t_f] \tag{1.1}$$

Dabei ist $t \in [t_0,t_f]$ und $F : [t_0,t_f] \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$.
Wir wollen den typischen Spezialfall einer *quasilinear-impliziten nichtlinearen DAE* betrachten:

$$\left.\begin{array}{rcl} A(t,x,z)\dot{x} & = & f(t,x,z) \\ 0 & = & g(t,x,z) \end{array}\right\} t \in [t_0,t_f]$$

A habe hierbei vollen Rang. Man nennt $x(t)$ die *differentiellen Zustände* und $z(t)$ die *algebraischen Zustände*.
Ein Spezialfall der linear-impliziten nichtlinearen DAE ist die *semi-explizite DAE*:

$$\left.\begin{array}{rcl} \dot{x} & = & f(t,x,z) \\ 0 & = & g(t,x,z) \end{array}\right\} t \in [t_0,t_f]$$

Oft ist es möglich, differentiell-algebraische Gleichungen durch analytische Differentiationen in gewöhnliche Differentialgleichungen überzuführen:

DEFINITION 1.1
Die Gleichung (1.1) hat den *differentiellen Index* m, wenn m die minimale Anzahl von analytischen Ableitungen

$$F(t,x,\dot{x}) = 0,\;\; \tfrac{\partial F(t,x,\dot{x})}{\partial t} = 0,\;\; \ldots,\;\; \tfrac{\partial^m F(t,x,\dot{x})}{\partial t^m} = 0 \tag{1.2}$$

ist, sodaß (1.2) es erlaubt, durch algebraische Manipulationen ein System gewöhnlicher Differentialgleichungen $\dot{x} = \zeta(x)$ zu extrahieren.

Bemerkung: Wenn $\partial g/\partial z$ regulär ist, so ist die semi-explizite DAE vom Index 1, denn nach einmaligem Ableiten nach t ergibt sich:

$$\begin{aligned} 0 &= g_t(t,x,z) + g_x(t,x,z)\dot{x} + g_z(t,x,z)\dot{z} \\ &\quad \text{bzw.} \\ \dot{z} &= g_z(t,x,z)^{-1}\left(g_t(t,x,z) + g_y(t,x,z)\dot{x}\right) \end{aligned}$$

Wir wollen dies im folgenden annehmen und nur DAEs vom Index 1 betrachten. Probleme von höherem Index behandelt man oft durch eine Index-Reduktion (z.B. durch Differentiation der algebraischen Gleichungen). Man kann aber auch Runge-Kutta-Methoden oder Mehrschritt-Methoden für Index-2-DAEs herleiten. Wir verweisen hier auf ([SBS98]), ([HW93]) und [Gea88].

Durch Anwendung des Satzes über implizite Funktionen lassen sich bei quasilinear-impliziten DAEs die algebraischen Variablen als Funktion der differentiellen Variablen schreiben: $z = \vartheta(x)$. Damit kann man die DAE in eine ODE umformen. Die sogenannte *Zustandsform* ergibt sich durch:

$$A(t,x,\vartheta(x))\dot{x} = f(t,x,\vartheta(x))$$

Dies ist von theoretischer Bedeutung, da sich nun alle Konvergenzeigenschaften und Verfahren von gewöhnlichen Differentialgleichungen übertragen lassen.

Um ein Anfangswertproblem zu definieren, müssen nur für die differentiellen Variablen Anfangswerte angegeben werden:

$$\left.\begin{aligned} A(t,x,z)\dot{x} &= f(t,x,z) \\ 0 &= g(t,x,z) \end{aligned}\right\} t \in [t_0,t_f]$$
$$x(t_0) = x_0$$

Die algebraischen Anfangswerte bestimmen sich aus der Konsistenzbedingung:

$$g(t_0,x_0,z(t_0)) = 0$$

Die Bestimmung konsistenter Anfangswerte kann mit globalisierten Newton-Verfahren oder Homotopie-Verfahren vorgenommen werden.

1.1.2 Lineare Mehrschritt-Verfahren

Zur Integration von differentiell-algebraischen Gleichungen vom Index 1 stehen verschiedene Lösungsmethoden zur Verfügung, die ursprünglich für gewöhnliche Differentialgleichungen (ODE) entwickelt wurden und deren Konvergenzeigenschaften sich auf differentiell-algebraische Gleichungen (DAE) übertragen lassen.
Die bekanntesten Methoden dürften wohl das Euler-Verfahren und die Runge-Kutta-Methoden sein. Darüber hinaus gibt es unter anderem noch Extrapolations-Methoden, Rosenbrock-Methoden und Mehrschritt-Methoden.

Für unsere Problemklasse sehr geeignet sind die *linearen Mehrschritt-Verfahren*, da sie wegen der Berücksichtigung früherer Funktionswerte der rechten Seite sehr effizient sind, was den Aufwand der Funktionsauswertungen angeht. Wir betrachten Sie der besseren Übersicht wegen in ihrer ursprünglichen Schreibweise für gewöhnliche Differentialgleichungen (ODE). Wie bei den Einschritt-Verfahren (z.B. Runge-Kutta-Methoden) wird eine Näherungslösung auf einem Gitter

$$t_{k+1} = t_k + h_k \quad t_k \in [t_0,t_f]$$

berechnet. Während man für die Theorie von einem äquidistanten Gitter ausgeht, ist in der Praxis eine geeignete Schrittweitensteuerung von großer Bedeutung. Im Gegensatz zu den Einschritt-Verfahren, bei denen zur Berechnung des Werts der Funktion an einem neuen Gitterpunkt η_{k+1} nur Informationen aus dem letzten Schritt η_k verwendet werden, greifen Mehrschritt-Verfahren auf Informationen aus mehreren vergangenen Schritten zurück.

Ein *lineares s-Schritt-Verfahren* ist gegeben durch eine Rekursion der Form:

$$\sum_{i=0}^{s} \alpha_i \eta_{n+i} = h \sum_{i=0}^{s} \beta_i f_{n+i} \tag{1.3a}$$

$$\text{wobei} \quad f_{n+i} := f(t_{n+i},\eta_{n+i}) \tag{1.3b}$$

Das Verfahren heißt *explizit*, wenn $\beta_s = 0$, sonst *implizit*.

Im folgenden wollen wir die Konvergenzeigenschaften der linearen Mehrschritt-Verfahren betrachten.

DEFINITION 1.2
Setzt man die exakte Lösung des Anfangswertproblems y statt der Gitterfunktion in das Diskretisierungsschema ein, so erhält man das Korrekturglied $\sigma := h\tau$:

$$\begin{aligned} \sum_{i=0}^{s} \alpha_i y(t+hi) &= h \sum_{i=0}^{s} \beta_i f_{n+i} + h\tau(t,h) \\ &= h \sum_{i=0}^{s} \beta_i f_{n+i} + \sigma(t,h) \end{aligned}$$

Man nennt σ den *lokalen Fehler* und τ den *Konsistenzfehler*.

DEFINITION 1.3
Das lineare Mehrschritt-Verfahren (1.3b) heißt *konsistent*, wenn gilt:

$$\begin{aligned} \bar{\tau}(h) &\to 0 \quad \text{für } h \to 0 \\ \text{mit} \quad \bar{\tau} &:= \max_{\tau \in [t_0,t_f]} \|\tau(t,h)\| \end{aligned}$$

Die *Konsistenzordnung* ist o, wenn gilt:

$$\bar{\tau}(h) = \mathcal{O}(h^o)$$

DEFINITION 1.4
Die *charakteristischen Polynome* eines Mehrschritt-Verfahrens sind definiert durch:

$$\begin{aligned} \rho(\xi) &:= \sum_{j=0}^{k} \alpha_j \xi^j \\ \sigma(\xi) &:= \sum_{j=0}^{k} \beta_j \xi^j \end{aligned}$$

Es gilt der folgende, leicht zu zeigende Satz:

SATZ 1.5 (KONSISTENZ VON LINEAREN MEHRSCHRITT-VERFAHREN)
Das lineare Mehrschritt-Verfahren (1.3b) ist genau dann konsistent, wenn gilt:

$$\begin{aligned} \rho(1) &= 0 \\ \rho'(1) - \sigma(1) &= 0 \end{aligned}$$

DEFINITION 1.6
Der *globale Fehler* wird definiert durch:

$$\varepsilon(t,h) := \eta_{k+1} - y(t_{k+1})$$

DEFINITION 1.7
Das Verfahren heißt *konvergent*, wenn gilt:

$$\begin{aligned} \bar{\varepsilon}(h) &\to 0 \quad \text{für } h \to 0 \\ \text{mit} \quad \bar{\varepsilon} &:= \max_{\varepsilon \in t_0,\ldots,t_N} \|\varepsilon(t,h)\| \end{aligned}$$

Die *Konvergenzordnung* ist o, wenn gilt:

$$\bar{\varepsilon}(h) = \mathcal{O}(h^o)$$

Stabilität bedeutet, daß kleine Störungen der Anfangswerte nicht stark fortgepflanzt werden und sich nicht stark auf die Lösung des Diskretisierungsverfahren auswirken. Die Betrachtung der Fehlerfortpflanzung ist wesentlich komplizierter als bei Einschritt-Verfahren, da der Fehler sich s-mal fortpflanzen kann.

DEFINITION 1.8
Ein lineares Mehrschritt-Verfahren heißt *stabil*, wenn es $k_1,k_2 < \infty$ unabhängig von h gibt mit

$$\bar{\varepsilon}(h) \leq k_1 \bar{\tau}(h) + k_2 \max_{i=0}^{s-1} \|\varepsilon_i\|$$

wobei ε_i die Fehler der notwendigen Startwerte $\eta_0, \ldots, \eta_{s-1}$ sind.

DEFINITION 1.9
Ein lineares Mehrschritt-Verfahren heißt *nullstabil*, wenn für die Nullstellen λ_j des charakteristischen Polynoms $\rho(\lambda)$ das sogenannte *Wurzelkriterium* gilt:

$$\begin{aligned} |\lambda_j| &\leq 1 \\ |\lambda_j| &= 1 \quad \Longrightarrow \lambda_j \text{ ist einfache Nullstelle von } \rho(\lambda) \end{aligned}$$

SATZ 1.10 (SATZ VON DAHLQUIST)
Sei $y \in \mathcal{C}^{o+1}$ und f lipschitz-stetig. Der Fehler der Startwerte sei 0, das heißt:

$$\eta_i \;=\; y(t_i) \quad i = 0, \ldots, k-1$$

Dann gilt:

1. *Ist das lineare Mehrschritt-Verfahren konsistent und nullstabil, dann ist es auch konvergent.*
2. *Ist die Konsistenzordnung o, so ist auch die Konvergenzordnung o.*

Der Beweis benutzt den Satz von Hirsch. Der Satz läßt sich auch in umgekehrter Richtung zeigen, sodaß gilt:

SATZ 1.11 (KONVERGENZ VON LINEAREN MEHRSCHRITT-VERFAHREN)
Ein lineares Mehrschritt-Verfahren ist genau dann konvergent, wenn es stabil und konsistent ist.

Insbesondere gilt dann:

$$\rho'(1) = \sigma(1) \neq 0$$

Die maximale Ordnung eines linearen Mehrschritt-Verfahrens ist $o = 2s + 1$. Wählt man die Koeffizienten so, daß möglichst viele Taylor-Terme verschwinden, ergibt sich eine maximale Ordnung von $o = 2s - 1$. Solche Verfahren sind allerdings nicht nullstabil.

SATZ 1.12 (ORDNUNG VON LINEAREN MEHRSCHRITT-VERFAHREN)
Ein nullstabiles s-Schritt-Verfahren hat höchstens die Ordnung

$$\begin{aligned} s+2 \quad &, \quad \textit{falls } s \textit{ gerade} \\ s+1 \quad &, \quad \textit{falls } s \textit{ ungerade} \\ s \quad &, \quad \textit{falls } \beta_s/\alpha_s < 0 \end{aligned}$$

DEFINITION 1.13
Das *Stabilitätsgebiet* eines linearen Mehrschritt-Verfahren ist definiert durch

$$S \;:=\; \left\{ z \in \mathbb{C} : \begin{array}{ll} |\xi_l| \leq 1 & \text{wenn } \xi_l \text{ einfache Wurzel von (1.3a)} \\ |\xi_l| < 1 & \text{wenn } \xi_l \text{ mehrfache Wurzel von (1.3a)} \end{array} \right.$$

DEFINITION 1.14
Ein lineares Mehrschritt-Verfahren heißt A-*stabil*, wenn

$$\mathbb{C}^- \subset S$$

wobei $\mathbb{C}^- := \mathbb{C} - \{x \in \mathbb{R}; x \leq 0\}$.

Es gilt allerdings:

SATZ 1.15 (ZWEITE DAHLQUIST-SCHRANKE)
Die maximale Konsistenzordnung eines A-stabilen linearen Mehrschritt-Verfahren ist 2.

Daher betrachte man eine Abschwächung der A-Stabilität, und zwar, daß nicht die gesamte negative Halbebene in $\mathcal{S}$ enthalten sein muß, sondern nur ein (möglichst großer) Winkelbereich:

DEFINITION 1.16
Ein lineares Mehrschritt-Verfahren heißt *A(α)-stabil*, wenn

$$S_\alpha := \{z \in \mathbb{C} : |\arg(-z)| \leq \alpha\} \subset \mathcal{S}$$

mit $\alpha \in [0; \pi/2]$

Beim Lösen von Anfangswertproblemen stellt man fest, daß für bestimmte Probleme explizite Methoden versagen. Man bezeichnet dies mit dem Begriff *Steifheit*. Ein System heißt *steif*, wenn die Jacobi-Matrix Eigenwerte mit negativen Realteilen hat, die stark unterschiedliche Größenordnungen haben. Die Zeitmaßstäbe, in denen sich die Zustände ändern, sind dann sehr unterschiedlich. In der Praxis bedeutet dies, daß explizite Methoden mit beschränktem Stabilitätsgebiet sehr kleine Schrittweiten verlangen. Für steife Systeme, wie sie z.B. bei chemischen Systemen oft auftreten, sind daher implizite Methoden zu verwenden.

1.1.3 Spezielle lineare Mehrschritt-Verfahren

Die meisten populären Verfahren sind *Integral-Verfahren*. In der Integralform

$$y(t_{k+1}) = y(t_i) + \int_{t_i}^{t_{k+1}} \dot{y}(t)dt$$

wird das Integral durch eine interpolatorische Quadratur-Formel ersetzt:

$$y(t_{k+1}) = y(t_i) + \int_{t_i}^{t_{k+1}} p(\tau; \dot{y}(t_{k+1}), \ldots, \dot{y}(t_{k+1-s}))d\tau + h_{k+1}\tau(t_k, h_{k+1})$$

Der Fehler $h\tau(t_k,h)$ ist der lokale Fehler der Quadratur, das heißt die Konsistenzordnung ist $\tau(t_k,h) = \mathcal{O}(h^{s+1})$ für das implizite Verfahren und $\tau(t_k,h) = \mathcal{O}(h^s)$ für das explizite. Das implizte Verfahren hat also eine höhere Konsistenzordnung als das explizite Verfahren, dafür muß in

jedem Schritt ein implizites Gleichungssystem gelöst werden (z.B. mit einer Funktionaliteration oder einem näherungsweisen Newton-Verfahren).

Für $i = k$ erhält man die *Adams-Verfahren*. Das *explizite Adams-Bashforth-Verfahren* läßt sich schreiben als:

$$\begin{aligned} \eta_{k+1} &= \eta_k + \int_{t_k}^{t_{k+1}} p(\tau; f_k, \ldots, f_{k+1-s}) d\tau \\ &= \eta_k + h \sum_{i=0}^{s-1} \beta_i f_{k+1-s+i} \end{aligned}$$

Das *implizite Adams-Moulton-Verfahren* läßt sich schreiben als:

$$\begin{aligned} \eta_{k+1} &= \eta_k + \int_{t_k}^{t_{k+1}} p(\tau; f_{k+1}, \ldots, f_{k+1-s}) d\tau \\ &= \eta_k + h \sum_{i=0}^{s} \beta_i f_{k+1-s+i} \end{aligned}$$

Alle impliziten Adams-Moulton-Verfahren sind nullstabil.

Eine Kombination aus beiden Verfahren ergibt sich im *Adams-Bashforth-Moulton-Verfahren* oder auch *Prädiktor-Korrektor-Verfahren*. Dabei liefert das Adams-Bashforth-Verfahren den Startwert für die Funktionaliteration zur Lösung der impliziten Gleichung innerhalb des Adams-Moulton-Verfahrens.

1. **Predict** Berechne η^0 aus dem Adams-Bashforth-Verfahren
2. **Evaluate** Berechne $f(t_{k+1},\eta^0) := f^0$
3. **Correct** Berechne $\eta^1 = h_k \beta_s f(t_{k+1},\eta^0) + \gamma_k$
4. **Evaluate** Berechne $f(t_{k+1},\eta^1) := f^1$

Danach wird $\eta_{k+1} = \eta^1$ und $f_{k+1} = f^1$ gesetzt und mit der nächsten Iteration fortgefahren.

Einen anderen Ansatz verfolgen die *BDF-Verfahren (Backward-Differentiation-Formulas)*. Hierbei faßt man das lineare Mehrschritt-Verfahren als Konstruktion eines Interpolationspolynoms auf. Man interpoliert dabei den neu zu berechnenden Punkt y_{k+1} und die bereits berechneten Punkte $y_k, \ldots, y_{k+1-s}$ und verlangt, daß das Interpolationspolynom $p(t; y_{k+1}, \ldots, y_{k+1-s})$ bei t_{k+1} die Differentialgleichung

$$\dot{p}(t_{k+1}; y_{k+1}, y_n, \ldots, y_{k-s}) = f(t_{k+1}, y_{k+1}, z_{k+1})$$

erfüllt. Der lokale Fehler ist gleich der Ableitung des Interpolationsfehlers, also ist die Konsistenzordnung s. Die BDF-Verfahren sind A-stabil für $s = 1,2$, $A(\alpha)$-stabil für $s \leq 6$ und instabil für $s > 6$ (siehe Abbildung 1.1). Der Beweis hierfür wurde zuerst von Cryer ([Cry72]) geführt.

s	1	2	3	4	5	6
α	$90°$	$90°$	$86.0°$	$73.4°$	$51.8°$	$17.8°$

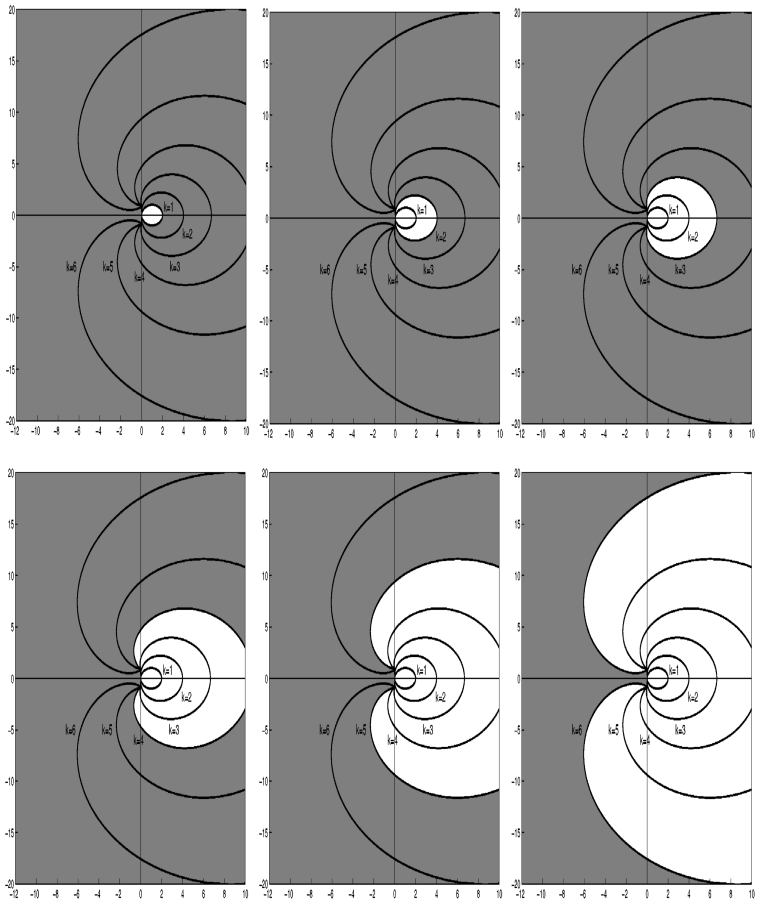

Abbildung 1.1: Stabilitätsgebiete (grau) der BDF-Verfahren für $k = 1, \ldots, 6$

Die Verfahren sind besonders geeignet für steife Systeme. Zwei weitere Vorteile der Verfahren sind, daß einerseits die BDF-Verfahren auch auf variablen Gittern formulierbar sind, und andererseits, daß aufgrund der Darstellung des Interpolationspolynoms durch Hinzufügen weiterer Summanden die Ordnung ohne größeren Aufwand erhöht werden kann, was eine simultane adaptive Schrittweiten- und Ordnungssteuerung möglich macht.
Am IWR wurde der auf dem BDF-Verfahren basierende DAE-Löser DAESOL ([Bau99]) bzw. DAESOL-II ([Alb06]) sowie das speziell auf mechanische Systeme zugeschnittene Integratorpaket MBSSIM , das verschiedene Adams-Bashforth-Moulton-Verfahren, Runge-Kutte-Verfahren

und Extrapolations-Verfahren beinhaltet ([vSW94]), entwickelt. Beide Integratoren wurden in Forschung und Industrieprojekten erfolgreich eingesetzt.

1.2 Problemformulierung und Lösungsmethoden

In diesem Abschnitt werden wir ein Problem der optimalen Steuerung formulieren, das auf der im vorhergehenden Abschnitt vorgestellten quasilinear-impliziten DAE beruht und die gebräuchlichsten Lösungsverfahren vorstellen.

1.2.1 Problemformulierung

Für die Optimierung führen wir Steuerfunktionen und Steuergrößen (Parameter) ein:

$$\left.\begin{array}{rcl} A(\cdot)\dot{x}(t) & = & f(x(t),z(t),u(t),p,t) \\ 0 & = & g(x(t),z(t),u(t),p,t) \end{array}\right\} t \in [t_0,t_f] \tag{1.4}$$

$x(t) \in \mathbb{R}^{n_x}$ bezeichne die differentiellen Zustände, $z(t) \in \mathbb{R}^{n_z}$ die algebraischen Zustände, $u \in \mathbb{R}^{n_u}$ die vektorwertige Steuerfunktion, $p \in \mathbb{R}^{n_p}$ die konstanten Parameter. Weiter nehmen wir wieder an, daß A regulär ist.
Zur Formulierung des Optimierungsproblems verwenden wir eine Zielfunktion (objective function/cost function) vom Bolza-Typ, die minimiert werden soll (z.B. Kosten):

$$\min_{x(t),z(t),u(t),p,t_f} \Phi(x(t_f),z(t_f),p,t_f) + \int_{t_0}^{t_f} \phi(x(t),z(t),u(t),p,t)dt$$

Φ wird Mayer-Term genannt, ϕ Lagrange-Term. Probleme mit Lagrange-Term in der Zielfunktion können behandelt werden durch eine geeignete Erweiterung der DAE (1.4) (siehe z.B. [Lei95]). In der verwendeten Software MUSCOD-II geschieht dies automatisch, wenn der Benutzer einen Lagrange-Term spezifiziert. Im folgenden werden wir zur einfacheren Darstellung nur Mayer-Zielfunktionale betrachten.
Weiterhin formulieren wir kontinuierliche nichtlineare Beschränkungen an Steuerungen und Zustände (z.B. obere und untere Schranken für die Steuerung)

$$h(x(t),z(t),u(t),p,t) \quad \geq \quad 0 \quad t \in [t_0,t_f]$$

sowie linear-gekoppelte Randbedingungen (z.B. zur Festlegung fester Anfangswerte)

$$r^s(x(t_0),z(t_0),p) + r^e(x(t_f),z(t_f),p) \left\{ \begin{array}{c} = \\ \geq \end{array} \right\} 0$$

Die letzteren dürfen sowohl Gleichungen als auch Ungleichungen enthalten.

Um allgemeinere Modelle mit Unstetigkeiten oder unterschiedlichen Modellphasen behandeln zu können, teilen wir den Zeithorizont $[t_0,t_f]$, in dem die Dynamik des Problems abläuft, in M Modellstufen ein:

$$[t_i,t_{i+1}], \quad i = 0,\ldots,M-1$$

Die Zeitpunkte der Stufenübergänge t_i (und damit die Dauer der einzelnen Modellstufen)werden zu Optimierungsvariablen. Die Dimensionen der Vektoren $x_i(t),z_i(t),u_i(t)$ sind n_i^x, n_i^z und n_i^u und können von Stufe zu Stufe verschieden sein.

Das entstehende mehrstufige Problem der optimalen Steuerung läßt sich zusammengefaßt wie folgt formulieren:

$$\min_{x_i(t),z_i(t),u_i(t),p,t_i} \sum_{i=0}^{M-1} \Phi_i(x_i(t_{i+1}),z_i(t_{i+1}),p,t_{i+1}) \tag{1.5a}$$

DAE-Modell-Stufen

$$\left.\begin{array}{rcl} A_i(\cdot)\dot{x}_i(t) & = & f_i(x_i(t),z_i(t),u_i(t),p,t) \\ 0 & = & g_i(x_i(t),z_i(t),u_i(t),p,t) \end{array}\right\} t \in [t_i,t_{i+1}], \quad i = 0,\ldots,M-1 \tag{1.5b}$$

Pfadbeschränkungen (Bedingungen an die Steuerungen und an die Zustände)

$$h_i(x_i(t),z_i(t),u_i(t),p,t) \geq 0, \quad t \in [t_i,t_{i+1}], \quad i = 0,\ldots,M-1 \tag{1.5c}$$

Stufenübergangsbedingungen

$$x_{i+1}(t_{i+1}) = c_i(x_i(t_{i+1}),z_i(t_{i+1}),p,t_{i+1}), \quad i = 0,\ldots,M-2 \tag{1.5d}$$

linear gekoppelte Randbedingungen

$$\sum_{i=0}^{M-1} (r_i^s(x_i(t_i),z_i(t_i),p,t_i) + r_i^e(x_i(t_{i+1}),z_i(t_{i+1}),p,t_{i+1})) \quad \left\{\begin{array}{c} = \\ \geq \end{array}\right\} \quad 0 \tag{1.5e}$$

DEFINITION 1.17
Erfüllt $T = (x(t),z(t),u(t),p)$ die Differentialgleichungen (1.5b), so heißt T Trajektorie. T heißt *zulässige Trajektorie*, wenn alle Gleichungen bzw. Ungleichungen von (1.5) erfüllt sind.

1.2.2 Lösungsmethoden für Probleme der optimalen Steuerung

Es gibt zahlreiche Methoden, Problem (1.5) zu lösen. Für einen guten Überblick über die gebräuchlichen Methoden empfehlen wir [BBB+01] und [Sag05].

Der klassische Ansatz benutzt das Pontryaginsche Maximumprinzip und führt zu den sogenannten *indirekten Methoden*. Die notwendigen Optimalitätsbedingungen werden benutzt, um das

Problem in ein Mehrpunkt-Randwertproblem umzuformen, das zum Beispiel mit Kollokation oder der Mehrziel-Methode gelöst werden kann. Das Randwertproblem enthält aber a-priori unbekannte Schaltpunkte, welche die Zeiten kennzeichnen, an denen eine Nebenbedingung aktiv oder inaktiv wird, was zu Sprüngen in den adjungierten Variablen führen kann. Der Hauptvorteil dieses Ansatzes ist die hohe Genauigkeit der erhaltenen Lösung, da auf eine Diskretrisierung der Steuerung verzichtet wird, und die Diskretisierung des unendlich-dimensionalen Problems erst nach der Formulierung der Optimalitätsbedingungen geschieht. Die Formulierung von Optimalitätsbedingungen in numerisch sinnvoller Weise, die wechselnde Schaltpunktstruktur und die notwendige Bereitstellung von Anfangswerten für alle Zustandsvariablen und alle adjungierten Variablen kennzeichnen die Nachteile dieses Ansatzes. Die Verfahren sind nur geeignet bei kleinen Problemen und einer hohen benötigten Genauigkeit.

Die *dynamische Programmierung* beruht auf dem Prinzip, daß jeder Teilbogen einer optimalen Trajektorie selbst optimal ist, sodaß das Problem der optimalen Steuerung in einzelne Teilprobleme zerlegt werden kann. Das Verfahren der dynamischen Programmierung besteht darin, zuerst die optimalen Lösungen der kleinsten Teilprobleme direkt zu berechnen, und diese dann geeignet zu einer Lösung eines nächstgrößeren Teilproblems zusammenzusetzen, und so weiter. Die für verschiedene Startwerte berechneten Ergebnisse der Teilprobleme mit kurzem Zeithorizont werden dabei in einer Tabelle gespeichert.
Läßt man die Zeithorizonte der Teilprobleme gegen Null gehen und führt quasi eine Dynamische Programmierung mit unendlich kleinen Zeitschritten durch, führt dies zu der *Hamilton-Jacobi-Bellman-Gleichung*. Man erhält dann die optimale Steuerung durch Lösen eines Randwertproblems bei dieser partiellen Differentialgleichung (partial differential equation, PDE).
Da der komplette Zustandsraum durchsucht wird, findet diese Methode sogar das globale Optimumg des Problems. Der Hauptnachteil ist jedoch, daß eine PDE in einem hochdimensionalen Zustandsraum gelöst werden muß. Aufgrund des sogenannten “Fluchs der Dimensionalität” (curse of dimensionality) ist diese Methode auf Systeme mit kleiner Zustands-Dimension (z.B. $2-3$) beschränkt. Neuere Entwicklungen umgehen den “Fluch der Dimensionalität” durch Ansätze mit neuronalen Netzen ([BT96]). Trotz alledem bleibt die Methode auf kleine Systeme beschränkt.

Die Grundidee der *direkten Methoden* ist es, das Problem (1.5) in ein endlich-dimensionales nichtlineares Problem (NLP) zu transformieren. Wir wollen drei Varianten vorstellen: das direkte Einfach-Schieß-Verfahren (direct single shooting), direkte Kollokation ([Str93], [Bie84]) und die direkte Mehrziel-Methode (direct multiple shooting). Bei allen dreien wird eine endlich-dimensionale Parametrisierung q der Steuerung u verwendet. Die Methoden unterscheiden sich in der Art und Weise, wie die Zustände behandelt werden, ob ein sequentieller Ansatz oder ein simultaner Ansatz verfolgt wird, bei dem die Lösung des Optimierungsproblems und die Lösung des Randwertproblems simultan erfolgen.

Beim *direkten Einfach-Schieß-Verfahren* werden die Zustände (x,z) als abhängige Variablen betrachtet. Das Problem (1.5) ist dann ein Problem in der endlich-dimensionalen Steuerung q, dem Anfangswert x_0 und den Parametern p. Das resultierende Optimierungsproblem hat wenige Freiheitsgrade, wenn die Dimension der Parameter q der diskretisierten Steuerungen klein ist und die

Anfangswerte fix sind. Dieses Verfahren ist einfach zu implementieren, da bereitstehende ODE- bzw. DAE-Löser und ein Standard-NLP-Löser (z.B. SQP-Verfahren) verwendet werden können.

Nachteile der Methode ergeben sich daraus, daß (1.5b) in Einem über das gesamte Integrationsintervall $[t_0,t_f]$ integriert wird. Es kann kein Vorwissen über die Zustandstrajektorien $(x(t),z(t))$ eingebracht werden. Man benötigt außerdem sehr gute Startwerte. Fehler, die durch ungenaue Eingangsdaten, Diskretisierung oder Rundung auftreten, können sich speziell bei DAEs mit Instabilitäten fortpflanzen und stark anwachsen. Falls keine globale Lipschitz-Konstante existiert, können Fehler der Anfangswerte bereits für $t < t_f$ "mit dem Faktor ∞ fortgepflanzt werden". Das heißt die Integration läuft in eine Singularität, in einen Pol hinein, und erreicht nicht den Endpunkt t_f. Außerdem wird in dem sequentiellen Verfahren in jedem Optimierungsschritt ein Anfangswertproblem mit hoher Genauigkeit gelöst (auch wenn die Steuerung noch weit weg von den optimalen Werten ist), was die Effizienz einschränkt.

Abhilfe schafft ein simultaner Ansatz, bei dem Lösung des Randwertproblems und Optimierung gleichzeitig geschehen. Wir stellen die beiden gebräuchlichsten Methoden, die direkte Kollokation und die direkte Mehrzielmethode vor.

Bei der *direkten Kollokation* basiert die Diskretisierung auf einer Approximation der Zustände durch ein stückweises Polynom auf einem Gitter. Um die Koeffizienten der Polynome zu bestimmen, verlangt man von der Lösung, daß sie die Differentialgleichung (1.5b) auf einer Unterteilung des Gitters erfüllt, nämlich auf den Kollokationspunkten, und daß sie an den Gitterpunkten stetig ist. Die Zustände sind also keine abhängigen Variablen mehr, sondern werden ebenfalls diskretisiert. Zielfunktion und Nebenbedingungen werden über die Werte an den Gitterpunkten definiert. Das entstehende NLP ist groß, aber strukturiert. Die Jacobi-Matrizen der Nebenbedingungen sind dünn besetzt. Das NLP kann mit einem geeigneten Verfahren, zum Beispiel durch die Innere-Punkt-Methode oder mit einem auf dünnbesetzte Probleme zugeschnittenen SQP-Verfahren gelöst werden. Die Vorteile der Methode sind, daß a-priori-Wissen über die Trajektorien eingebracht werden kann und daß die Methode im Gegensatz zu sequentiellen Methoden eine geringe Fehlerfortpflanzung beinhaltet. Der Nachteil der Methode ist, daß eine Adaptivität des Zeitgitters, was speziell für steife Probleme entscheidend ist, nicht so einfach zu realisieren ist, da sich dadurch die Dimensionen des zu Grunde liegenden NLP verändern.

Eine Methode, welche die Nachteile des Einfach-Schieß-Verfahrens und der Kollokation vermeidet, ist die *direkte Mehrziel-Methode*, die wir im nächsten Abschnitt vorstellen möchten. Der wesentliche Unterschied zu anderen simultanen Methoden ist, daß die Lösung der Differentialgleichungen immer noch durch Integration geschieht und moderne, fehlerkontrollierte DAE-Integratoren verwendet werden können. Abbildung 1.2 zeigt eine Übersicht über die vorgestellten Methoden.

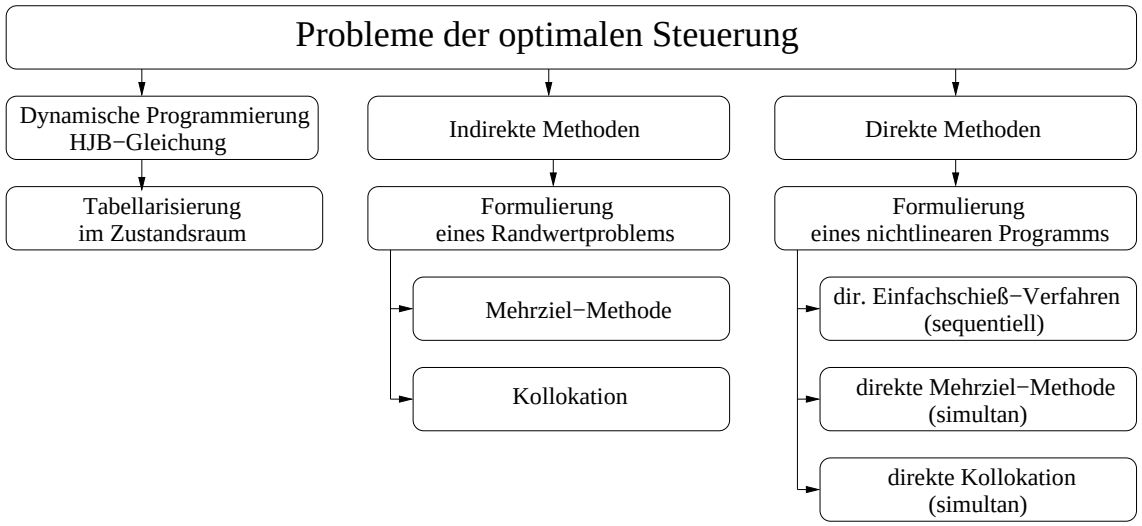

Abbildung 1.2: Übersicht: Methoden für Probleme der optimalen Steuerung

1.3 Diskretisiertes Problem – Bocks direkte Mehrziel-Methode

In diesem Abschnitt stellen wir *Bocks direkte Mehrziel-Methode (Bock's direct multiple shooting method)* ([BP84]) vor, die das unendlich-dimensionale kontinuierliche Problem der optimalen Steuerung in vier Schritten in ein strukturiertes endlich-dimensionales NLP transformiert:

- Zeittransformation
- Diskretisierung der Steuerung
- Parametrisierung der Zustände
- Diskretisierung der Nebenbedingungen

Wir verwenden eine spezielle Variante der direkten Mehrziel-Methode (direkt multiple shooting), die von Bock und Plitt ([Pli81], [BP84]) und in verallgemeinerter Form von Schulz, Leineweber et al. ([SBS98], [LBBS03]) vorgestellt wurde und seitdem angewendet und weiterentwickelt wird ([Die98], [FMT02], [LBBS03], [BP04], [TBK04], [Sch04]).

Die Methode wurde in dieser Form in dem Softwarepaket MUSCOD-II ([Lei99], [LBBS03]) implementiert, das auch als Basis für den Multiple-Setpoint-Algorithmus verwendet wurde, den wir in Abschnitt 3.6 präsentieren werden.

1.3.1 Zeittransformation

Um Modelle mit variablen Stufendauern lösen zu können, führen wir auf jeder Modellstufe $i = 1,\ldots,M-1$ die Zeittransformation

$$\theta_i(t,v) \quad = \quad t_i + \tau d_i \quad \tau \in [0,1] \tag{1.6}$$

ein mit

$$\begin{aligned} t_i &= t_0 + \sum_{k=0}^{i-1} d_k \\ v &:= (t_0,d_0,\ldots,d_{M-1}) \end{aligned}$$

und betrachten nunmehr nur noch einen fixen Zeithorizont $[0,1]$ in jeder Stufe.

1.3.2 Stückweise Diskretisierung der Steuerungen

Die Steuerungen haben eine unendliche Anzahl an Freiheitsgraden. Um sie in eine endlichdimensionale Form zu bringen, wählen wir auf jeder Stufe ein festes, dimensionsloses Diskretisierungsgitter:

$$0 \quad = \quad \tau_{i,0} < \tau_{i,1} < \ldots < \tau_{i,m_i} \quad = \quad 1 \tag{1.7}$$

Dann wählen wir eine stückweise Approximation $\hat{u}_i$ der Steuerung u (siehe Abbildung 1.3):

$$\hat{u}_{ij}(\tau) \quad := \quad \chi_{ij}(\tau,q_{ij}), \quad \tau \in I_{ij} := [\tau_{ij},\tau_{i,j+1}], \quad i = 1,\ldots,M-1 \quad j = 0,\ldots,m_i-1$$

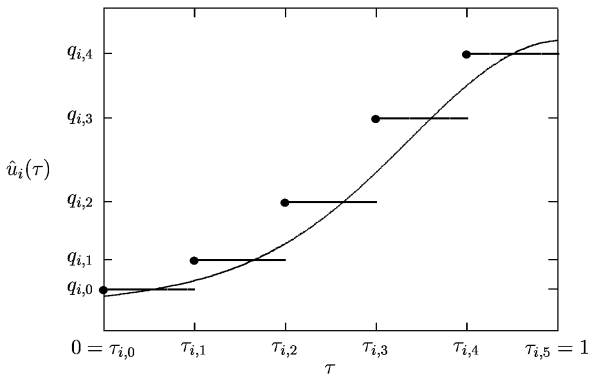

Abbildung 1.3: Stückweise konstante Approximation der Steuerungsfunktion

Für die Optimierung werden nun die lokalen Parameter q_{ij} verwendet. χ_{ij} sind festgelegte lokale Basisfunktionen. Am gebräuchlichsten ist die Verwendung von Vektoren von einfachen Polynomen, zum Beispiel konstanten oder linearen Funktionen. Für eine stückweise konstante Approximation ergibt sich:

$$\chi_{ij} = q_{ij}$$

Für eine stückweise lineare Approximation ergibt sich

$$\chi_{ij}(\tau,q_{ij}) \;=\; q_{ij}^1 + \frac{\tau - \tau_{ij}}{\tau_{i,j+1} - \tau_{ij}}(q_{ij}^2 - q_{ij}^1), \qquad q_{ij} = \begin{pmatrix} q_{ij}^1 \\ q_{ij}^2 \end{pmatrix}$$

durch eine Interpolation der Randwerte des Intervalls q_{ij}^1 und q_{ij}^2. Die Diskretisierung sowie die Anzahl und Lage der Stüzpunkte kann auf jeder Stufe individuell gewählt werden.
Ist Stetigkeit der Steuerungs-Approximationen erwünscht, können zusätzlich Stetigkeitsbedingungen eingeführt werden.
Um gute Diskretisierungen zu erhalten ist eine geeignete Wahl der Anzahl und Position der Gitterpunkte in (1.7) nötig. In der Praxis beginnt man oft mit einem etwas gröberen Gitter und verfeinert das Gitter sukzessive in geeigneter Weise, bis man eine zufriedenstellende Approximation erhält. Algorithmen zur adaptiven Verfeinerung der Gitter finden sich unter anderem in [SBD$^+$05] und [Sch05].
Obwohl auch globale Diskretisierungen der Steuerungen vorstellbar sind, werden in der Praxis meist stückweise Diskretisierungen verwendet, um unnötige nichtlineare Kopplungen im diskretisierten Problem zu vermeiden und unstetige Steuerungen (z.B. *Bang-Bang-Steuerungen*) einfach behandeln zu können. Wir werden in Abschnitt 2.3.2 sehen, welche Strukturen sich durch diese Diskretisierung ergeben und wie diese ausgenutzt werden können.

1.3.3 Parametrisierung der Zustände

Wir unterteilen das Integrationsintervall in Teilintervalle, indem wir das Diskretisierungsgitter (1.7) verwenden, führen an den Gitterpunkten τ_{ij} zusätzliche Optimierungsparameter s_{ij}^x, s_{ij}^z ($i = 0,\ldots,M-1$, $j = 0,\ldots,m_i$) ein und lösen ein System von entkoppelten Anfangswertproblemen:

$$\left.\begin{array}{rcl} A_i(\cdot)dx_i(\tau)/d\tau &=& f_i(x_i(\tau),z_i(\tau),\chi_{ij}(\tau,q_{ij}),p,\theta_i(\tau,v))d_i \\ 0 &=& g_i(x_i(\tau),z_i(\tau),\chi_{ij}(\tau,q_{ij}),p,\theta_i(\tau,v)) \\ && -\alpha_{ij}(\tau)g_i(s_{ij}^x,s_{ij}^z,\chi_{ij}(\tau_{ij},q_{ij}),p,\theta_i(\tau_{ij},v)) \end{array}\right\} \tau \in I_{ij}$$
$$\left.\begin{array}{rcl} x_i(\tau_{ij}) &=& s_{ij}^x \\ z_i(\tau_{ij}) &=& s_{ij}^z \end{array}\right\} \text{Anfangswerte} \qquad (1.8)$$

Es wird eine relaxierte Fassung der originalen DAE aus (1.5b) verwendet mit einem Dämpfungsterm (ähnlich einer Homotopie-Strategie), um inkonsistente algebraische Anfangswerte zuzulassen und dadurch eine effiziente Lösung der DAE zu ermöglichen. Der Dämpfungsfaktor $\alpha_{ij}(\tau)$

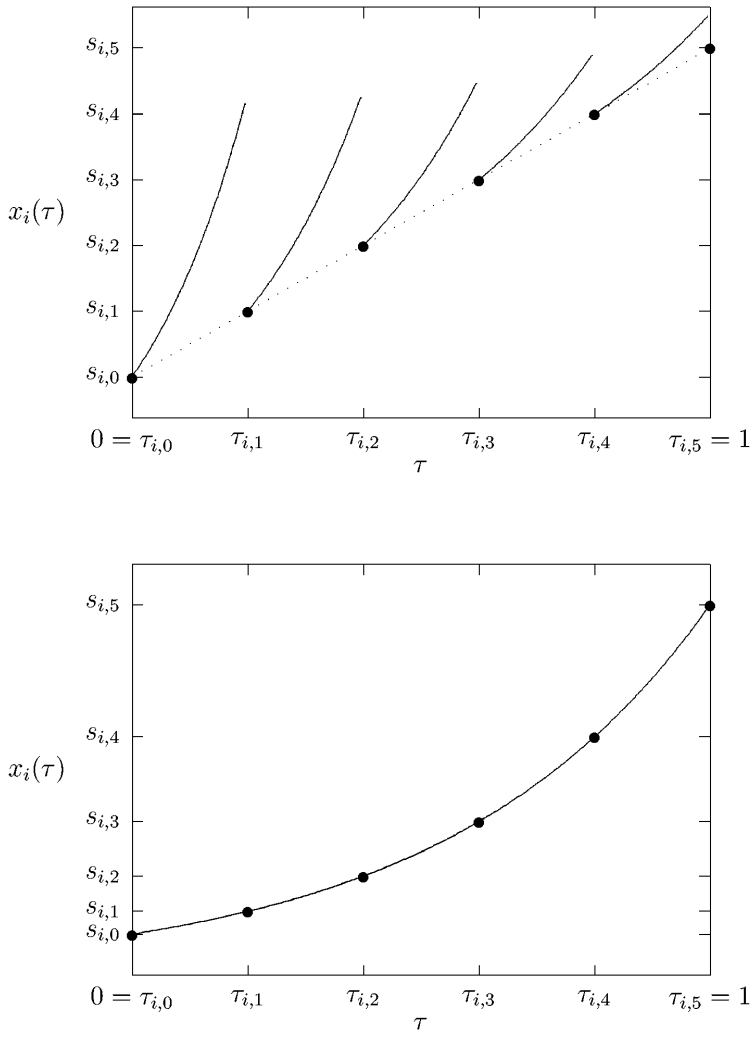

Abbildung 1.4: Direkte Mehrziel-Methode: Anfangstrajektorie und optimale Lösung
(In diesem Beispiel wurde die Anfangstrajektorie durch lineare Interpolation der Randwerte bestimmt)

Lösung des Randwertproblems und Optimierung geschehen gleichzeitig, erst in der Lösung sind die Randwertbedingungen und die Stetigkeitsbedingungen erfüllt.

kann eine beliebige nicht-wachsende, nichtnegative Funktion auf I_{ij} sein, die $\alpha_{ij}(\tau_{ij}) = 1$ erfüllt. Implementiert in MUSCOD-II ist die folgende Variante:

$$\alpha_{ij}(\tau) \;=\; \exp\left(-\bar{\alpha}\frac{\tau - \tau_{ij}}{\tau_{i,j+1} - \tau_{ij}}\right) \qquad \bar{\alpha} \geq 0$$

Die Lösungen der Anfangswertprobleme sind voneinander unabhängige Trajektorien auf $[\tau_{i,j},\tau_{i,j+1}]$. Es bezeichne $x_i(\tau_{i,j+1}; s^x_{ij},s^z_{ij},q_{ij},p,v)$ die differentiellen Zustandsvariablen zur Zeit $\tau = \tau_{i,j+1}$, die durch Integration von (1.8) erhalten werden. Damit die Lösung des NLPs die originale DAE (1.5b) erfüllt, formuliert man zusätzliche Stetigkeitsbedingungen (matching conditions)

$$x_i(\tau_{i,j+1}; s^x_{ij},s^z_{ij},q_{ij},p,v) - s^x_{i,j+1} \;=\; 0 \quad j = 1,\ldots,m_i - 1 \tag{1.9}$$

und algebraische Konsistenzbedingungen (algebraic consistency conditions)

$$g_i(x_i(\tau),z_i(\tau),\chi_{ij}(\tau_{ij},q_{ij}),p,\theta_i(\tau_{ij},v)) \;=\; 0 \quad j = 1,\ldots,m_i \tag{1.10}$$

Zur Berechnung der Zustände $x_i(\tau_{i,j+1}; s^x_{ij},s^z_{ij},q_{ij},p,v)$ und deren Sensitivitäten werden effiziente, adaptive DAE-Löser benutzt.
Setzen wir nun die unabhängigen Zustands-Trajektorien $x_i(\tau)$, $z_i(\tau)$ in den Lagrange-Term in (1.5) ein, lassen sich diese Teile der Zielfunktion simultan berechnen.
Die beiden Nebenbedingungen (1.9), (1.10) beseitigen die zusätzlichen Freiheitsgrade, die durch die Einführung der Parameter s^x_{ij}, s^z_{ij} eingeführt wurden. Während der Optimierung ist es keineswegs notwendig, daß die Bedingungen erfüllt sind – im Gegenteil, es ist eine entscheidende Eigenschaft der direkten Mehrziel-Methode, daß unzulässige Startlösungen behandelt werden können. Zur Lösung des NLPs wird ein sogenannter *Infeasibile-Path-Algorithmus* verwendet, das heißt die iterierten Zustandsvariablen im Lösungsprozeß sind unzulässig bezüglich der DAE-Diskretisierung (1.8), die Lösung des Optimierungsproblem und die Lösung des Randwertproblems geschieht simultan. Die Verwendung der relaxierten DAE und damit das Zulassen von inkonsistenten Zustandsvariablen an den Mehrzielknoten ist außerdem wichtig, wenn Unstetigkeiten an den Punkten des Diskretisierungsgitter entstehen, wie zum Beispiel durch stückweise konstante Steuerungen. Ein *Feasible-Path-Algorithmus* würde eine konsistente Initialisierung des DAE-Modells nach jeder Unstetigkeit erfordern, was bei große Modellen sehr aufwendig werden kann.
Auf den ersten Blick mag die Einführung von zusätzlichen Parametern s^x_{ij},s^z_{ij} und zusätzlichen Gleichungen ungünstig erscheinen, aber aufgrund der entkoppelten Struktur des entstehenden Optimierungsproblems lassen sich sehr effiziente Lösungsalgorithmen entwickeln (siehe [Boc81], [BP84], [Boc87]). Weiterhin sind die entstehenden Algorithmen aufgrund der Unterteilung in Mehrzielintervalle gut zur Parallelisierung geeignet ([BPDLP04],[GB94]).

1.3.4 Diskretisierung der kontinuierlichen Nebenbedingungen

Die Nebenbedingungen an die Steuerungen und die Zustände sind kontinuierlich und müssen ebenfalls diskretisiert werden. In der Regel beschränkt man sich darauf, die Einhaltung der Ne-

benbedingungen auf einem Zeitgitter zu fordern. Gegebenfalls kann man das Gitter an den kritischen Stellen sukzessive verfeinern, bis die gewünschten Ergebnisse erzielt werden können. Eine Möglichkeit ist es, das bereits eingeführte Mehrzielgitter (1.7) zu verwenden:

$$h_i(s_{ij}^x,s_{ij}^z,\chi_{ij}(\tau_{ij},q_{ij}),p,\theta_i(\tau_{ij},v)) \quad \geq \quad 0 \quad j=0,\ldots,m_i \quad i=0,\ldots,M-1$$

Strikte Erfüllung der Nebenbedingungen (strict feasibility) läßt sich durch den Einsatz bestimmter Techniken erreichen, zum Beispiel durch Transformation jeder Pfadbeschränkung zu einer äquivalenten Endbedingung an eine zusätzlich eingeführte Hilfsvariable ([SS78], [VSP94]) oder durch die sogenannte *Maximum Violation Tracking Method* ([Pot06]).

1.3.5 Diskretisierte Problemformulierung

Die Diskretisierungen der letzten Abschnitte führen zu folgendem nichtlinearen Problem (NLP):

$$\min_{s_{ij}^x,s_{ij}^z,q_{ij},p,v} \sum_{i=0}^{M-1} \Phi(s_{i,m_i}^x,s_{i,m_i}^z,p,\theta_i(\tau_{i,m_i},v)) \tag{1.11a}$$

unter den folgenden Nebenbedingungen:

Stetigkeitsbedingungen

$$x(t_{i,j+1};s_{ij}^x,s_{ij}^z,q_{ij},p,v) - s_{i,j+1}^x = 0 \quad j=0,\ldots,m_i-1 \quad i=0,\ldots,M-1 \tag{1.11b}$$

Konsistenzbedingungen

$$g(s_{ij}^x,s_{ij}^z,\chi_{ij}(\tau_{ij},q_{ij}),p,\theta_i(\tau_{ij},v)) = 0 \quad j=0,\ldots,m_i \quad i=0,\ldots,M-1 \tag{1.11c}$$

Diskretisierte Steuerungs- und Zustandbeschränkungen

$$h(s_{ij}^x,s_{ij}^z,\chi_{ij}(\tau_{ij},q_{ij}),p,\theta_i(\tau_{ij},v)) \geq 0 \quad j=0,\ldots,m_i \quad i=0,\ldots,M-1 \tag{1.11d}$$

Stufenübergangsbedingungen

$$c_i(s_{i,m_i}^x,s_{i,m_i}^z,p,\theta_i(\tau_{i,m_i},v)) - s_{i+1,0}^x = 0 \quad i=0,\ldots,M-2 \tag{1.11e}$$

Linear gekoppelte Mehrpunktbedingungen

$$\sum_{i=0}^{M-1}\sum_{j=0}^{m_i} r_{ij}(s_{ij}^x,s_{ij}^z,p,\theta_i(\tau_{ij},v)) \left\{ \begin{array}{c} = \\ \geq \end{array} \right\} 0 \tag{1.11f}$$

Die Berechnung von $x(t_{i,j+1};s_{ij}^x,s_{ij}^z,q_{ij},p,v)$ erfordert hierbei die Lösung des relaxierten Anfangswertproblems (1.8). Die Gitterpunkte τ_{ij} sowie die Zeittransformation θ_i seien definiert wie in (1.6) bzw. (1.7).

Die linear gekoppelten Bedingungen wurden in dieser Formulierung erweitert von Randbedingungen zu allgemeinen Mehrpunktbedingungen, das heißt auch Zustände an Multiple-Shooting-Knoten im Innern einer Stufe können in die Nebenbedingung eingehen.

Eine weitere leichte Modifikation des diskretisierten Problems ist in der Praxis sinnvoll: Die globalen Modell-Parameter p,v müssen "lokalisiert" werden durch Einführen von neuen lokalen Variablen p_{ij},v_{ij} sowie zusätzlichen Kopplungs-Nebenbedingungen:

$$\left.\begin{array}{rcl} p_{ij} & = & p_{0,0} \\ v_{ij} & = & v_{0,0} \end{array}\right\} \quad \forall (i,j) \neq (0,0) \tag{1.11g}$$

Die zusätzlichen Nebenbedingungen ändern nicht viel an der Komplexität des Problems. Bei der Lösung des quadratischen Programms innerhalb der Lösung des Optimierungsproblems können

die zusätzlichen Variablen und Nebenbedingungen in einem sogenannten *Vorkondensierungs-Schritt* eliminiert werden (siehe Abschnitt 2.3.4). Wie wir später zeigen werden, ist die Lokalisierung der Parameter wichtig, um eine partiell separierbare Lagrange-Funktion (siehe (2.13)) zu erreichen. Bei der Formulierung Strukturen zu schaffen, ist wichtig, um effiziente Algorithmen entwickeln zu können. Aus diesem Grund sind in der Formulierung des nichtlinearen Problems allgemeine nichtlineare gekoppelte Mehrpunktbedingungen ebensowenig erlaubt wie eine kontinuierlich-stetige (nicht-stückweise) Formulierung der Steuerungen.
Ein weiterer wichtiger Punkt bei der praktischen Lösung des Problems (1.11) ist die Skalierung der Variablengrößen, die großen Einfluß auf die Performance der Algorithmen hat. Für geeignete Skalierungsstrategien für unsere Algorithmen verweisen wir auf [BC85], [Bie84] und [Lei95].

2 Lösung von nichtlinearen Problemen mit dem SQP-Verfahren

In diesem Kapitel stellen wir mit dem SQP-Verfahren (Sequential Quadratic Programming method) eine effiziente Lösungsmethode für beschränkte nichtlineare Optimierungsprobleme vor. Auf Basis des SQP-Verfahrens werden wir in Abschnitt 3.6 einen Algorithmus für Multiple-Setpoint-Probleme bei Problemen der optimalen Steuerung vorstellen.
Im ersten Abschnitt werden die notwendigen Optimalitätsbedingungen (Kuhn-Tucker-Bedingungen) für nichtlineare Probleme formuliert, auf denen das SQP-Verfahren basiert. Im zweiten Abschnitt werden die Details der SQP-Verfahren, speziell auch neuere Ansätze, beleuchtet. Im dritten Abschnitt beschreiben wir ein SQP-Verfahren, das speziell auf Probleme der optimalen Steuerung zugeschnitten ist, bevor wir im letzten Abschnitt kurz die Software MUSCOD-II vorstellen, in der der Algorithmus als Kernroutine implementiert ist.

2.1 Grundlegendes

In diesem Abschnitt formulieren wir die Bedingungen für ein Optimum des nichtlinearen Optimierungsproblems, auf dem das SQP-Verfahren beruht, und stellen die Grundidee des SQP-Verfahrens vor.

2.1.1 Optimalitätsbedingungen

Ein allgemeines nichtlineares Optimierungsproblem mit Gleichungs- und Ungleichungsnebenbedingungen läßt sich wie folgt formulieren:

$$\begin{array}{lll} \min\limits_{w \in D} & F(w) & \\ G(w) & = & 0 \\ H(w) & \geq & 0 \end{array} \qquad \begin{array}{l} F : D \subseteq \mathbb{R}^n \to \mathbb{R} \\ G : D \subseteq \mathbb{R}^n \to \mathbb{R}^l \\ H : D \subseteq \mathbb{R}^n \to \mathbb{R}^m \end{array} \qquad (l < n \text{ i.a.}) \tag{2.1}$$

Im allgemeinen geht man von dreimal stetig differenzierbaren Funktionen aus (f,g,$h \in \mathcal{C}^3(D)$). Wir benutzen folgende gebräuchliche Schreibweisen: $G_i(w)$, $H_i(w)$ seien die i.ten Zeilen von $G(w)$ bzw. $H(w)$. Die Gradienten $\nabla F(w)$, $\nabla G_i(w)$ und $\nabla H_i(w)$ der skalaren Funktionen F, G_i und H_i seien als Spaltenvektoren definiert. Weiterhin sei

$$\nabla G \;=\; (\nabla G_1, \ldots, \nabla G_l) \quad , \quad \nabla H \;=\; (\nabla H_1, \ldots, \nabla H_m)$$

das heißt, die Jacobi-Matrizen von G bzw. H sind gegeben als $\nabla G(w)^T$ bzw. $\nabla H(w)^T$. Die Hesse-Matrizen von F, G_i und H_i seien bezeichnet mit $\nabla^2 F(w)$, $\nabla^2 G_i(w)$ und $\nabla^2 H_i(w)$.

Zur Formulierung der Optimalitätsbedingungen benötigen wir einige Definitionen.

DEFINITION 2.1
Die *zulässige Menge* $S \subset \mathbb{R}^n$ ist definiert durch:

$$S \;:=\; \{w \in D \mid G(w) = 0, H(w) \geq 0\}$$

Ein Punkt $w \in S$ heißt *zulässiger Punkt*.

DEFINITION 2.2
$w^* \in S$ heißt *globales Minimum* von (2.1), wenn gilt:

$$F(w^*) \leq F(w) \quad \forall\, w \in S$$

$w^* \in S$ heißt *lokales Minimum*, wenn es eine Umgebung $U \subset D$ von w^* gibt mit:

$$F(w^*) \leq F(w) \quad \forall\, w \in U \cap S$$

Minima heißen *strikt*, falls gilt:

$$F(w^*) < F(w) \quad \forall\, w \in S \text{ bzw. } \forall\, w \in U \cap S$$

Im folgenden betrachten wir Theorie und Algorithmen für lokale Minima. Algorithmen zum Auffinden globaler Minima haben einen erheblich höheren Rechenaufwand, der bei größeren Problemen zu inakzeptablen Rechenzeiten führt, zumal in der Praxis eine lokale Optimierung oft ausreichend ist. Methoden zur globalen Optimierung von dynamischen Systemen finden sich zum Beispiel in [PA04].

DEFINITION 2.3
Sei $w \in S$ (zulässiger Punkt)

1. $I(w) := \{i \mid H_i(w) = 0\} = \{i_1, \ldots, i_s\}$ heißt *Indexmenge der aktiven Ungleichungen* oder *Active-Set*.
2. $I^{\perp}(w) := \{i \mid H_i(w) > 0\}$ heißt *Indexmenge der inaktiven Ungleichungen*
3. $\tilde{H} := (H_{i_1}, \ldots, H_{i_s})$
   $\tilde{H} : D \subseteq \mathbb{R}^n \longrightarrow \mathbb{R}^s$
4. $\tilde{G} := (G, \tilde{H})$
   $\tilde{G} : D \subseteq \mathbb{R}^n \longrightarrow \mathbb{R}^{l+s}$
5. $w \in S$ heißt *regulär*, wenn $\nabla \tilde{G}(w)^T$ vollen Rang $l + s \leq n$ hat.
   O.B.d.A. seien die ersten $l+s$ Spalten von $\nabla \tilde{G}(w)^T$ linear unabhängig und v entsprechend aufgeteilt in $v = (v', v'')$. Dann läßt sich die Menge $\tilde{S} := \{v \mid \tilde{G}(v) = 0\}$ in einer Umgebung U von w durch eine Abbildung ϑ parametrisieren, d.h. $\exists \vartheta$ mit $\tilde{G}(v', \vartheta(v')) = 0$.

Zulässigkeit kann nur erhalten werden durch Bewegung entlang eines nichtlinearen Pfades im $\mathbb{R}^n$, dessen Punkte die Gleichungen und Ungleichungen erfüllt. Aktive Ungleichungen können hierbei evtl. inaktiv werden. Inaktive Ungleichungen erlauben jedwede hinreichend kleine Störung und sind aus lokaler Sicht daher von keiner Bedeutung.

DEFINITION 2.4
$\mathcal{L}(w,\mu,\lambda) := F(w) - \lambda^T G(w) - \mu^T H(w)$
mit $\lambda \in \mathbb{R}^l, \mu \in \mathbb{R}^m$ heißt *Lagrange-Funktion*.

DEFINITION 2.5
$T(w^*) := \{p \in \mathbb{R}^n \mid \nabla_w \tilde{G}(w^*)p = 0\}$ heißt *Tangentialraum*.

DEFINITION 2.6
$T^+(w^*) := \{p \in \mathbb{R}^n \mid \nabla_w G(w^*)p = 0, \nabla_w H_i(w^*)p = 0 \quad \forall\, i \in I(w^*) \text{ und } \mu_i^* > 0\} \supset T(w^*)$

Mit diesen Definitionen lassen sich jetzt die notwendigen bzw. hinreichenden Optimalitätsbedingungen formulieren.

SATZ 2.7 (NOTWENDIGE OPTIMALITÄTSBEDINGUNGEN)
Sei w^* regulär und lokales Minimum, dann $\exists \lambda^* \in \mathbb{R}^l, \mu^* \in \mathbb{R}^m$ mit:

1. *Notwendige Optimalitätsbedingungen 1. Ordnung:*

$$\begin{aligned} \nabla_w \mathcal{L}(w^*,\lambda^*,\mu^*) &= 0 \quad \textit{d.h.} \nabla F(w^*) = \nabla G(w^*)\lambda^* + \nabla H(w^*)\mu^* && \text{(2.2a)} \\ \mu^* &\geq 0 && \text{(2.2b)} \end{aligned}$$

   und es gilt die Komplementaritätsbedingung:

$$\mu_j^{*T} H_j(w^*) = 0 \quad \forall j = 1,\ldots,m \qquad \text{(2.2c)}$$

2. *Notwendige Optimalitätsbedingungen 2. Ordnung:*

$$p^T \nabla_w^2 \mathcal{L}(w^*,\lambda^*,\mu^*)p \geq 0 \quad \forall\, p \in T(w^*) \qquad (2.3)$$

   d.h. die Hesse-Matrix ist positiv semidefinit auf dem Tangentialraum.

SATZ 2.8 (HINREICHENDE OPTIMALITÄTSBEDINGUNGEN)
Sei $w^* \in S$ und erfülle die notwendigen Optimalitätsbedingungen 1.Ordnung. Weiter sei erfüllt:

$$p^T \nabla_w^2 \mathcal{L}(w^*,\lambda^*,\mu^*)\, p > 0 \quad \forall\, 0 \neq p \in T^+(w^*)$$

Dann ist w^* striktes lokales Minimum.

Die notwendige Optimalitätsbedingungen aus Satz 2.7 heißen auch *Kuhn-Tucker-Bedingungen* oder *Karush-Kuhn-Tucker-Bedingungen (KKT-Bedingungen)*. Ein Punkt, der die KKT-Bedingungen erfüllt, heißt *KKT-Punkt*.
Eine andere Interpretation von (2.2) ist, daß der Gradient der Zielfunktion $\nabla F(w^*)$ als Linearkombination der Gradienten der Nebenbedingungen $\nabla G_i(w^*)$, $\nabla H_i(w^*)$ geschrieben werden kann.
Die Lagrange-Multiplikatoren μ_i^*, die sich auf inaktive Ungleichungen beziehen, sind Null aufgrund der Komplementaritätsbedingung (2.2c). Die Lagrange-Multiplikatoren, die sich auf Gleichungsnebenbedingungen oder aktive Ungleichungsnebenbedingungen beziehen, lassen sich als sogenannte *Schattenpreise* interpretieren: Sei F^* der optimale Wert des ursprünglichen Problems und $F_{\delta_i}^*$ der optimale Wert des Problems, das entsteht, wenn man die Ungleichung $G_i(w) = 0$ durch $G_i(w) = \delta_i$ ersetzt, dann läßt sich zeigen:

$$\lim_{\delta_i \to 0} \frac{F_{\delta_i}^* - F^*}{\delta_i} = \lambda_i^*$$

2.1.2 Grundidee des SQP-Verfahrens

Die Grundidee beim SQP-Verfahren (Sequential Quadratic Programming method) ist es, die Lösung eines allgemeinen nichtlinearen Problems (NLP) zu ersetzen durch die Lösung einer Folge von geeigneten einfacheren Problemen. Indem sukzessive quadratische Approximationen des Problems formuliert und gelöst werden, wird eine Technik entwickelt, um in einem iterativen Prozeß einen KKT-Punkt des NLPs zu finden. Da Terme höherer Ordnung vernachlässigt werden, sind die quadratischen Modelle nur in einer Umgebung der aktuellen Iterierten verläßliche Approximationen des ursprünglichen Problems und es muß eine Liniensuche oder eine Trust-Region-Technik angewendet werden, um eine vernünftige neue Iterierte zu finden. SQP-Verfahren zählen zu den leistungsfähigsten bekannten Verfahren zur Lösung von beschränkten nichtlinearen Optimierungsproblemen.

Ausgehend von einem beschränkten NLP in Standardformulierung

$$\begin{aligned} \min_{w \in D} & F(w) \\ G(w) &= 0 \\ H(w) &\geq 0 \end{aligned}$$

und einer Startlösung w_0 ergibt sich eine SQP-Iteration

$$w_{k+1} = w_k + t_k \underbrace{\Delta w_k}_{:= y_k}$$

aus der Lösung des folgenden quadratischen Programms:

$$\min_{y_k \in \Omega_k} \tfrac{1}{2} y_k^T \nabla_w^2 \mathcal{L}_k y_k + \nabla_w F_k^T y_k \tag{2.4}$$

$$\begin{aligned} G_k + \nabla_w G_k^T y_k &= 0 \\ H_k + \nabla_w H_k^T y_k &\geq 0 \end{aligned} \tag{2.5}$$

Dabei haben wir folgende abkürzende Schreibweisen verwendet:

$$\begin{aligned} F_k &:= F(w_k) \\ G_k &:= G(w_k) \\ H_k &:= H(w_k) \\ \mathcal{L}_k &:= \mathcal{L}(w_k,\lambda_k,\mu_k) \end{aligned}$$

Setzt man in die Zielfunktion die exakten Hesse-Matrizen der Lagrange-Funktion $\nabla_w^2\mathcal{L}(w,\lambda,\mu)$ ein, führt dies auf ein Newton-Verfahren für die Kuhn-Tucker-Bedingungen und Zulässigkeit, das lokal quadratisch konvergent ist (Äquivalenz von SQP-Verfahren und Newton-Verfahren). Setzt man Approximationen $B_k \approx \nabla_w^2\mathcal{L}(w,\lambda,\mu)$, die aus Update-Formeln gewonnen werden ein, erhält man die Quasi-Newton-SQP-Verfahren. Für geeignete Näherungen von $\nabla_w^2\mathcal{L}(w,\lambda,\mu)$ konvergiert das SQP-Verfahren lokal linear. $B_k = I$ führt auf das Gradientenverfahren. Verwendet man geeignete Update-Formeln, erhält man lokal superlineare Konvergenz. Der Beweis hierfür (siehe z.B. [Rie01]) benutzt die Äquivalenz von Newton-Verfahren und SQP-Verfahren sowie den lokalen Kontraktionssatz ([Boc87]). Globale Konvergenz kann erreicht werden, indem man eine Schrittweiten-Strategie für t_k oder eine Trust-Region-Technik für Ω_k einführt (siehe Abschnitt 2.2.4). Für weitere Details zur Konvergenz von SQP-Verfahren verweisen wir auf [Rob74], [Wri97] und [DS83].

2.2 Das SQP-Verfahren im Detail

Das SQP-Verfahren wurde als erstes von Wilson 1963 ([Wil63]) entwickelt. Eine erste richtungsweisende Implementierung stammt von Han und Powell ([Pow78]) Ende der 70er Jahre. Seitdem wurden viele Modifikationen und Varianten vorgeschlagen. In den folgenden Abschnitten werden wir die Details beleuchten, an denen sich verschiedene SQP-Verfahren unterscheiden können und die verschiedenen Varianten vorstellen. Speziell behandelt werden die Wahl einer Approximation der Hesse-Matrix der Lagrange-Funktion, die Lösung des gleichungs- und ungleichungsbeschränkten quadratischen Programms, der Einsatz von Strategien zur Erhaltung von globaler Konvergenz und die Verwendung von reduzierten SQP-Verfahren bei großen nichtlinearen Optimierungsproblemen.

2.2.1 Approximation der Hesse-Matrix

Die Wahl der Approximation der Hesse-Matrix der Lagrangrefunktion kann großen Einfluß auf das Verhalten der Algorithmus haben. Ebenso wichtig wie geeignete Update-Formeln ist es, eine geeignete Startapproximation zu finden.

2.2.1.1 Anfangsapproximation B_0

Als Startapproximation für die Hesse-Matrix wird in vielen Implementierungen der SQP-Methode die Einheitsmatrix verwendet. Es zeigt sich jedoch, daß diese Wahl selbst für gut-

skalierte Probleme zu unverhältnismäßig großen Schritten Δw_k in den ersten Iterationen führen kann. Mit einer Schrittweitenbegrenzung in den ersten Iterationen läßt sich zwar verhindern, daß der Algorithmus komplett fehlschlägt, aber die Richtung des Schritts bleibt die gleiche.
Bessere Richtungen in den ersten Iterationen werden durch eine Strategie erreicht, die erstmals von Plitt ([Pli81]) vorgeschlagen wurde. Ähnlich wie bei einer Trust-Region-Strategie wird erst eine vernünftige Beschränkung des ersten Schritts berechnet und dann ein modifiziertes quadratisches Programm gelöst, das garantiert, daß der Schritt der optimale Schritt unter der gegebenen Beschränkung ist. So bleibt man in der Umgebung von w_0, in der die Linearisierungen der Nebenbedingungen geeignete Approximationen der Nebenbedingungen sind.
Um eine vernünftige Schranke für die Norm des ersten Schritts zu finden, definiert man ein sogenanntes *Least Distance QP* $Q_{\mathcal{R}}(w_0)$:

$$\begin{aligned} &\min_{\Delta w} \|\Delta w\|_2^2 \\ G(w_0) + \nabla G(w_0)^T \Delta w &= 0 \\ H(w_0) + \nabla H(w_0)^T \Delta w &\geq 0 \end{aligned}$$

Die Lösung $\Delta w_0^{\mathcal{R}}$ ist ein Schritt von w_0 zum "nächsten" Punkt, der auf den linearisierten Beschränkungen liegt.
Plitt zeigt nun für den gleichungsbeschränkten Fall, daß sich bei einer Wahl von

$$B_0 = \frac{1}{\kappa_{\mathcal{R}}} I$$

mit

$$\kappa_{\mathcal{R}} = \sqrt{M_{\mathcal{R}}^2 - 1}$$

eine Beschränkung des Schritts

$$\|\Delta w_0\|_2 \leq M_{\mathcal{R}} \|\Delta w_0^{\mathcal{R}}\|_2$$

erreicht wird, wobei $M_{\mathcal{R}}$ eine wählbare empirische Konstante ist (z.B. $M_{\mathcal{R}} = 2$).
Um zu verhindern, daß $\|\Delta w_0^{\mathcal{R}}\|_2$ und $\|\nabla F(w_0)\|_2$ zu beschränkt werden und Werte nahe bei Null annehmen, modifiziert man die Formel in praktischen Umsetzungen zu

$$\kappa_{\mathcal{R}} = \sqrt{M_{\mathcal{R}}^2 - 1}\ \frac{\max(\|\Delta w_0^{\mathcal{R}}\|_2, \epsilon_{\mathcal{R}} \|w_0\|_2)}{\max(\|\nabla F(w_0)\|_2, \epsilon_{\mathcal{R}})}$$

Diese Vorgehensweise läßt sich auch für den ungleichungsbeschränkten Fall anwenden. Da die quadratischen Programme $Q_{\mathcal{R}}(w_0)$ und $Q(w_0, B_0)$ nicht unbedingt das gleiche Active-Set haben müssen, ist diese Vorgehensweise in gewisser Hinsicht heuristisch.

2.2.1.2 Update-Formeln für B_k

Um superlineare Konvergenz zu erhalten, muß für die aus den Updates berechneten Approximationen B_{k+1} der Hesse-Matrix gelten:

$$\left(B_{k+1} - \nabla^2 F(w_{k+1})\right) \frac{w_{k+1} - w_k}{\|w_{k+1} - w_k\|} \longrightarrow 0$$

Oft verwendete Update-Formeln sind der Broyden-Update B^B und der Symmetrische Broyden-Update B^{SB}, beides Rang-1-Updates, sowie die Rang-2-Updates BFGS-Update (Broyden, Fletcher, Goldfarb und Shanno) B^{BFGS} und die *DFP-Formel* (Davidon-Fletcher-Powell) B^{DFP} (Rang-2-Updates). Die entsprechenden Formeln lauten:

$$\begin{aligned}
B^B &:= B_k + \frac{(q - B_k p)p^T}{p^T p} \\
B^{BS} &:= B_k + \frac{(q - B_k p)(q - B_k p)^T}{(q - B_k p)^T p} \\
B^{BFGS} &:= B_k + \frac{qq^T}{q^T p} - \frac{rr^T}{p^T r} \\
B^{DFP} &:= B_k + \frac{(q - r)q^T + q(q - r)^T}{q^T p} - \frac{(q - r)^T pqq^T}{(q^T q)^2}
\end{aligned}$$

wobei, bei zum Zweck der besseren Übersichtlichkeit bei p_k, q_k, r_k der Index k weggelassen wurde. Es gilt hierbei:

$$\begin{aligned}
(p =)\ p_k &:= w_{k+1} - w_k \\
(q =)\ q_k &:= \nabla_w \mathcal{L}(w_{k+1}, \lambda_{k+1}, \mu_{k+1}) - \nabla_w \mathcal{L}(w_k, \lambda_{k+1}, \mu_{k+1}) \\
(r =)\ r_k &:= B_k p_k
\end{aligned}$$

Die Formeln erfüllen folgende Eigenschaften:

- Alle Update-Formeln erfüllen die *Sekantenbedingung*:

  $$B_{k+1} p_k = q_k \quad \text{(folgt aus } \nabla F(w_{k+1}) = \nabla F(w_k) + \nabla^2 F(w_k) p_k + \ldots\text{)}$$

- B^B erfüllt die Broyden-Formel

  $$B_{k+1} y = B_k y \quad \forall\, y \perp p,$$

  ist aber nicht symmetrisch.

- B^{SB} ist symmetrisch, aber nicht notwendigerweise positiv definit.
  (Genauer gesagt gilt: Ist $(q - B_k p)^T p > 0$ so folgt, daß B^{BS} positiv definit ist.)

- Ist B_k positiv definit und $q^T p > 0$, dann sind B^{BFGS} und B^{DFP} positiv definit.

- Alle Verfahren mit einem der beiden Rang-1-Updates konvergieren lokal superlinear, wenn gilt
  - w_0 ist nahe genug an der Lösung w^*
  - $B_0 = \nabla^2 F(w_0)$
  - $t_k = 1$
- Verfahren mit einem der beiden Rang-2-Updates konvergieren sogar global, wenn mit exakter Liniensuche gearbeitet wird, beim BFGS-Update genügt approximative Liniensuche.

Powell schlug in seiner Implementierung eine modifizierte BFGS-Update-Formel vor. Statt der normalen BFGS-Formel

$$B_{k+1} \;=\; B_k + U(B_k,p,q)$$

verwendete er die folgende modifizierte Formel:

$$\begin{aligned} B_{k+1} &= B_k + U(B_k,p,\eta) \\ \text{mit} \quad \eta &:= \Theta q + (1-\Theta) B_k p \quad 0 < \Theta < 1 \end{aligned}$$

Θ wird dabei so gewählt, daß η der Vektor ist, der am nächsten an q ist, der die Bedingung

$$p^T \eta \geq \epsilon_\Theta p^T B_k p \;>\; 0$$

erfüllt. ϵ ist dabei ein empirischer Faktor $\epsilon \in [0.1,0.2]$ (siehe [Pow85]). Θ hat also den Wert:

$$\Theta \;=\; \begin{cases} 1 & \text{wenn } p^T q \geq \epsilon_\Theta p^T B_k p \\ \dfrac{(1-\epsilon_\Theta) p^T B_k p}{p B_k p - p^T q} & \text{sonst} \end{cases}$$

B_k bleibt so positiv definit, auch wenn $\nabla^2_w \mathcal{L}_k$ oder $\nabla^2_w \mathcal{L}(w^*,\lambda^*,\mu^*)$ indefinit sind.

Für eine große Anzahl von Problemen funktioniert diese Strategie gut, jedoch zeigt sich bei einigen Problemen, daß die modifizierte BFGS-Formel zu stark schlecht-konditionierten Matrizen führen kann, was zu schlechten Suchrichtungen führen kann. ([Pli81],[Pow85]) In diesen Fällen empfiehlt sich eine alternative Update-Stragie, die sogenannten Limited-Memory-Updates. Hierbei wird nur eine gewisse Anzahl $\bar{l}$ (z.B. $\bar{l} = 5$ oder $\bar{l} = 10$) der vorangegangenen Update-Vektoren gespeichert anstatt der Updatematrizen selbst. Folgende Limited-Memory-Repräsentation der Hesse-Matrix-Approximation ersetzt dann (2.13):

$$(B_{ij})_{k+1} \;=\; \begin{cases} \frac{1}{\kappa_\mathcal{R}} I_{ij} + \sum_{l=0}^{k} \left(\dfrac{(\eta_{ij})_l (\eta_{ij})_l^T}{(c_{ij})_l} - \dfrac{(\zeta_{ij})_l (\zeta_{ij})_l^T}{(d_{ij})_l} \right) & 0 \leq k < \bar{l} \\ \frac{1}{\kappa_\mathcal{R}} I_{ij} + \sum_{l=k-\bar{l}+1}^{k} \left(\dfrac{(\eta_{ij})_l (\eta_{ij})_l^T}{(c_{ij})_l} - \dfrac{(\zeta_{ij})_l (\zeta_{ij})_l^T}{(d_{ij})_l} \right) & k \geq \bar{l} \end{cases}$$

mit

$$\begin{aligned}(\zeta_{ij})_l &:= (B_{ij})_l(p_{ij})_l \\ (\eta_{ij})_l &:= (\Theta_{ij})_l(q_{ij})_l + (1-(\Theta_{ij})_l)(\zeta_{ij})_l \\ (c_{ij})_l &:= (\eta_{ij})_l^T(p_{ij})_l \\ (d_{ij})_l &:= (p_{ij})_l(\zeta_{ij})_l\end{aligned}$$

Die Matrizen $(B_{ij})_l$ werden nie explizit aufgestellt, sodaß der Speicherbedarf signifikant reduziert wird, wenn die Matrix eine ausgeprägte Blockstruktur besitzt und l nicht zu groß ist. Wichtiger noch ist, daß die Kondition der Matrix kontrolliert werden kann durch Verkleinern der Zahl $\bar{l}$ der letzten Updates, die berücksichtigt werden sollen, weil der perfekt konditionierte Term $1/\kappa_{\mathcal{R}}$ mehr umd mehr dominiert. Dies zeigt auch, daß eine gute Wahl für die Anfangsapproximation in diesem Verfahren noch wichtiger ist als beim Standard-SQP-Verfahren. Für mehr über Limited-Memory-Updates empfehlen wir [Noc80].

Eine weitere Alternative ist die Approximation der Hesse-Matrix durch finite Differenzen. Da die einzelnen Blöcke simultan gestört werden können, ist die Zahl der zusätzlichen Auswertungen der Lagrange-Gradienten gleich der Größe des größten Blocks in der Hesse-Matrix. Die Genauigkeit der entstehenden zweiten Ableitungen ist zwar nicht sehr hoch, aber während Funktionsauswertungen und Gradienten sehr genau bestimmt werden müssen, braucht die Hesse-Matrix nicht mit hoher Genauigkeit vorzuliegen, sodaß die Methode sich als zufriedenstellend herausgestellt hat. Verglichen mit den Update-Strategien steigt die Konvergenzrate und geht in die Nähe von quadratischer Konvergenz, wie wir sie bei Verwendung einer exakten Hesse-Matrix haben. Gerade bei großen Systemen steigt aber der zusätzliche Rechenaufwand enorm.

2.2.2 Berechnung von Ableitungen durch Automatisches Differenzieren

Gewöhnlich geschieht die Berechnung der Gradienten durch finite Differenzen, das heißt durch Vorwärts-Approximation

$$\nabla F(w) := \frac{F(w+\eta_j e_j) - F(w)}{\eta_j}$$

wobei e_j j.ter Einheitsvektor und η_j eine kleine Zahl – oder durch zentrale Approximation

$$\nabla F(w) := \frac{F(w+\eta_j e_j) - F(w-\eta_j e_j)}{2\eta_j}$$

wobei e_j j.ter Einheitsvektor und η_j eine kleine Zahl.
Kritisch ist hierbei die Wahl von η_j. Nur für $\eta_j \to 0$ wird die korrekte Ableitung berechnet. Aber kleine η_j führen zu numerischen Schwierigkeiten aufgrund von Auslöschungs-Fehlern. Umgekehrt führen zu große η_j zu Diskretisierungsfehlern. Selbst im besten Fall ist die Genauigkeit der

Ableitungen ε_A nur $\varepsilon_A = \sqrt{\varepsilon_F}$, wenn ε_F die Genauigkeit der Funktionswerte ist. Noch schlechter wird das Verfahren bei Berechnungen von zweiten Ableitungen.

Effizienter ist die automatische Erzeugung von Ableitungen (*Automatisches Differenzieren*) ([Gri00]). Die Idee des automatischen Differenzierens ist, Programmcode einer Funktion f zu untersuchen und durch Erstellen eines Ausführungsgraphen (computational graph) in elementare Operationen zu zerlegen, deren Auswertung Zwischenergebnisse liefert. Die Berechnung der Ableitung der Ausgangsvariablen (Endergebnisse) $y := F(w)$ nach den Eingangsvariablen x kann nun auf zwei Varianten geschehen. In der ersten Variante werden in einem *Vorwärtsmodus* sukzessive die Ableitungen der Zwischenergebnisse nach den Eingangsvariablen berechnet. Der Aufwand ist unabhängig von der Zahl der Ausgangsvariablen

$$\text{Aufwand}(F') \quad \leq \quad (1 + 1.5n) * \text{Aufwand}(F)$$

wobei n die Dimension der Eingangsvariablen ist.
Im *Rückwärtsmodus* arbeitet man den Ausführungsgraph in umgekehrter Reihenfolge ab und berechnet sukzessive die Ableitungen der Ausgangsvariablen nach den Zwischenergebnisse. Die Komplexität ist nun unabhängig von der Zahl der Eingangsvariablen

$$\text{Aufwand}(F') \quad \leq \quad (1.5 + 2.5m) * \text{Aufwand}(F)$$

wobei m die Dimension der Ausgangsvariablen ist.

2.2.3 Lösung des quadratischen Programms

Ein entscheidender Schritt beim Lösen des nichtlinearen Problems (1.11) ist die Lösung des quadratischen Problems $Q(w_k,B_k)$.

2.2.3.1 Unbeschränkter Fall

$$\min \quad \frac{1}{2} y^T B y + g^T y$$

Ist B positiv definit, dann ergibt sich die eindeutige Lösung durch $By + g = 0$, andernfalls ist das Problem unbeschränkt. Lösungsmethoden sind zum Beispiel Cholesky-Zerlegung oder Konjugierte-Gradienten-Verfahren.

2.2.3.2 Gleichungsbeschränkter Fall

$$\begin{aligned} \min \quad & \tfrac{1}{2} y^T B y + g^T y \\ Cy + c \quad &= \quad 0 \end{aligned}$$

Wir gehen von folgenden Voraussetzungen aus:

- $C \in \mathbb{R}^{l \times n}$ hat Vollrang.
- $p^T B p > 0 \quad \forall\, 0 \neq p \in \{q \mid Cq = 0\}$

Dann lauten die Kuhn-Tucker-Bedingungen:

$$\exists \lambda : \begin{pmatrix} B & C^T \\ C & 0 \end{pmatrix} \begin{pmatrix} y \\ \lambda \end{pmatrix} = -\begin{pmatrix} g \\ c \end{pmatrix}$$

Die Matrix ist regulär, symmetrisch, aber indefinit. Das heißt, das Cholesky-Verfahren ist nicht anwendbar. Lösungsmöglichkeiten sind Konjugierte-Gradienten-Verfahren, Nullraum-Methode oder bei positiv definitem B Bildraum-Methode ([NW99]).

2.2.3.3 Gleichungs- und ungleichungsbeschränkter Fall

$$\begin{array}{lrll} \min & \frac{1}{2} y^T B y + g^T y & & \\ & Cy + c & = & 0 \qquad (l \text{ Gleichungen}) \\ & Dy + d & \geq & 0 \end{array}$$

Das gebräuchlichste Lösungsverfahren benutzt die sogenannte *Active-Set-Strategie*. Ziel ist es, die Menge der aktiven Indices I zu bestimmen (siehe Definition 2.3), um die Ungleichungsnebenbedingungen in aktive und inaktive Nebenbedingungen einzuteilen. Für ein "eingefrorenes" Active-Set werden die aktiven Ungleichungsnebenbedingungen wie Gleichheitsnebenbedingungen behandelt. Algorithmus 2.1 zeigt, wie durch sukzessives Lösen von quadratischen Programmen mit "eingefrorenem" Active-Set die Lösung eines gleichungs- und ungleichungsbeschränktem quadratischen Programm gefunden werden kann.

ALGORITHMUS 2.1 (ACTIVE-SET-STRATEGIE)

1. $k := 0$
   Bestimme einen zulässigen Punkt $y_0 \in S$
   und die zugehörige Menge $I_0 := \{i \mid D_i y_0 + d_i = 0\}$
   Voraussetzung ist $\mid I_0 \mid + l \leq n$

2. Löse $QP(I_k)$:

$$\begin{aligned} \min \tfrac{1}{2} y^T B y + g^T y & \\ Cy + c &= 0 \\ D_i y + d &= 0 \qquad \forall\, i \in I_k \end{aligned}$$

   und erhalte Lösung $\hat{y}$

3. Falls $\hat{y} \in S$ setze $y_{k+1} := \hat{y}$, $I_{k+1} := I_k$ und gehe zu Schritt 7.
   Ansonsten weiter

4. Setze $\Delta y := \hat{y} - y_k$ und betrachte die Gerade $y_k + t\Delta y$ für $0 \leq t \leq 1$
   Bestimme das kleinste t, für das eine Ungleichung j verletzt wird:

   a) Setze das Residuum $r_i(t) := D_i(y_k + t\Delta y) + d_i \quad \forall i \notin I_k$
      Die i.te Ungleichung wird dann nicht verletzt für $t \leq t_i$, wobei

$$t_i = -\frac{D_i y_k + d_i}{D_i \Delta y}$$

   b) Setze $\bar{t} := \min_i \; t_i$

5. Führe die Iteration durch

$$\begin{aligned} y_{k+1} &:= y_k + \bar{t}\Delta y \\ I_{k+1} &:= I_k \cup \{j\} \end{aligned}$$

6. Falls $\mid I_{k+1} \mid + l = n$, setze $k := k + 1$ und gehe zu Schritt 7.
   Sonst setze $k := k + 1$ und gehe zu Schritt 2.

7. Wir haben eine Lösung y_k von $QP(I_k)$ erreicht
   Berechne jetzt die zugehörigen Lagrange-Multiplikatoren λ_k, μ_k.

8. Falls alle $(\mu_k)_i \geq 0 \quad (i \in I_k)$, ist eine Lösung des QPs mit Ungleichungen gefunden $\Longrightarrow$ FERTIG
   Ansonsten gibt es ein $(\mu_k)_\xi < 0$
   Aus der Theorie folgt, daß die Lösung des QPs für $I_k - \{\xi\}$ ein besseres Zielfunktional hat
   Also $I_{k+1} = I_k - \{\xi\}$ und gehe zu 1)
   (Nur einen Index entfernen, wähle z.B. den kleinsten Index oder den Index mit betragsgrößtem $(\mu_k)_i$)

2.2.4 Globale Konvergenz

Die Konvergenzsätze für SQP-Verfahren machen nur Aussagen über lokale Konvergenzeigenschaften, das heißt Konvergenz in einer geeigneten Umgebung der Lösung. Tatsächlich zeigt sich in der Praxis, daß diese Umgebung üblicherweise nicht groß genug ist, um die Anfangswerte zu Beginn der Optimierung einzuschließen. Indefinite projizierte Hesse-Matrizen und damit unbeschränkte quadratische Progamme können auftreten. Dies macht Globalisierungsstrategien in SQP-Verfahren unverzichtbar. Ziel ist es, globale Konvergenz zu erzwingen, ohne die lokale superlineare Konvergenzrate zu beeinflussen. Die beiden populärsten Globalisierungstrategien sind die Liniensuche sowie die Trust-Region-Technik; neuere Ansätze bieten die Filter-Methoden oder der Restrictive Monotonicity Test.

Wir benutzen eine Liniensuche basierend auf der Watchdog-Technik ([CLPP82]) und mit einer Standardgütefunktion und heuristischen Gewichtungsfaktoren ([Pow78]) sowie alternativ eine Boxstep-Trust-Region-Technik ([DS83], [Fle87]).

Wir betrachten zunächst Methoden der *Liniensuche*, bevor wir uns dann mit der Wahl einer geeigneten *Gütefunktion* auseinandersetzen. Anschließend diskutieren wir das Problem des Maratos-Effekts und stellen Techniken wie die *Watchdog-Technik* und die *Second-Order-Correction* vor, die verhindern, daß bei der Liniensuche unter Umständen die Konvergenzrate beeinträchtigt wird. Abschließend stellen wir als Alternativen zur Liniensuche die *Trust-Region-Technik* und die *Filter-Methoden* vor. Für den Restrictive Monotonicity Test verweisen wir auf [BKS00].

2.2.4.1 Liniensuche

Für die *Liniensuche (line search)* wird eine *Gütefunktion (merit function)* T benötigt, die Verbesserungen sowohl im Zielfunktional als auch in der Zulässigkeit mißt. Mit der Wahl einer geeigneten Gütefunktion beschäftigen wir uns in nächstem Abschnitt. Die Aufgabe der Liniensuche ist es dann, die Gütefunktion ausgehend von w_k entlang der Abstiegsrichtung d_k zu minimieren:

$$\begin{aligned} w_{k+1} &:= w_k + t_k \Delta w_k \\ \text{mit} \qquad t_k &:= \min_{t \in (0,\alpha]} T(w_k + t_k \Delta w_k) \end{aligned}$$

Da eine exakte Liniensuche in jedem Iterationsschritt viel zu aufwendig wäre, muß man auch hier geeignete Heuristiken aufstellen, die einerseits in jedem Schritt einen möglichst guten Abstieg der Zielfunktion garantieren, andererseits mit wenig Rechenaufwand auskommen. Dafür gibt es verschiedene Strategien (siehe z.B. [DS83]).
Zur Vereinfachung der Schreibweise definieren wir für diesen Abschnitt $\rho(t) := T(w_k + t\Delta w_k)$.

Goldstein-Regeln :

- Regel 1 :
  $\rho(t_k) \leq \rho(0) + t_k\mu\rho'(0)$ mit einem $0 < \mu < 1$ (z.B. $\mu = 0.1$)

- Regel 2 :
  $\rho(t_k) \geq \rho(0) + t_k\gamma\rho'(0)$ mit einem $\mu < \gamma < 1$

Armijo-Goldstein-Strategie :

- Regel 1 :
  wie oben

- Regel 2 :
  Wähle t_k als größte Zahl einer Folge $(\sigma\zeta^i)$, wobei $0 < \sigma \leq 1$ und $0 < \zeta < 1$, mit:
  $\rho(t_k) \leq \rho(0) + t_k\mu\rho'(0)$ mit einem $0 < \mu < 1$.
  (z.B. $\sigma := 1$, $\zeta \in [\frac{1}{10}, \frac{1}{2}]$, $\mu := 0.1$)

Powell-Strategie :

- Regel 1 :
  wie oben

- Regel 2 :
  $\rho'(t_k) \geq \sigma\rho'(0)$ mit $\sigma \in [0.1, 0.9]$
  $|\rho'(t_k)| \leq -\sigma\rho'(0)$

Am gebräuchlichsten ist die Variante nach Armijo-Goldstein. Es gilt folgender Satz:

SATZ 2.9 (SCHITTKOWSKI)
Sei $N(w_0) := \{w \in D \mid T(w) \leq T(w_0)\}$ kompakt, sei t_k gemäß den Armijo-Goldstein-Regeln gewählt, und gelte

$$\nabla T(w_k)^T \Delta w_k \;\leq\; -\varepsilon^{Sch} \|\Delta w_k\| \; \|\nabla T(w_k)\|$$

mit einem $\varepsilon^{Sch} > 0$. Dann gilt: Jeder Häufungspunkt w^* ist stationärer Punkt, das heißt $\nabla T(w^*) = 0$.

Ein Algorithmus zur Bestimmung der Schrittweite hat dann folgende Form:

ALGORITHMUS 2.2 (SCHRITTWEITENBESTIMMUNG)

1. $t_0 := 1, \;\; k := 0$
2. Falls $\rho(t_k) < \rho(0) + t_k \rho'(0)\mu \Longrightarrow$ STOP
   ansonsten weiter
3. Nähere $\rho(t)$ durch
$$\tilde{\rho}(t) \;\; := \;\; (\rho(t_k) - \rho(0) - t_k\rho'(0))t^2 + \rho'(0)t + \rho(0)$$
4. Wähle $\tilde{t}$ so, daß $\rho(\tilde{t}) = 0$
$$\tilde{t} \;\; := \;\; \frac{\rho'(0)}{2(t_k\rho'(0) - (\rho(t_k) - \rho(0))}$$
5. $t_{k+1} := \max\{\beta t_k, \tilde{t}\}$ mit $\beta \in [0.2, 0.5]$
6. $k := k + 1$, gehe zu Schritt 2.

2.2.4.2 Gütefunktion

Die *Gütefunktion (merit function)* (oder auch Niveaufunktion oder Straffunktion) soll Verbesserungen sowohl im Zielfunktional als auch in der Zulässigkeit messen. Eine geeignete Gütefunktion ist zum Beispiel:

$$\begin{aligned} T^1(w) \;\; &:= \;\; F(w) + \sum_{i=1}^{l} \alpha_i \mid G_i(w) \mid + \sum_{i=1}^{m} \beta_i \mid \bar{H}_i(w) \mid \\ \text{mit } \bar{H}_i(w) \;\; &:= \;\; \min(0, H_i(w)) \end{aligned}$$

Sie ist verträglich mit dem SQP-Verfahren, wie das folgende Lemma (siehe [Fle87]) zeigt:

LEMMA 2.10
Falls in einem Punkt w_k das quadratische Programm mit positiv definiter Hesse-Matrix B_k die Lösung $\Delta w_k, \lambda_k, \mu_k$ liefert und die Bedingungen

$$\begin{aligned} \mid (\lambda_k)_i \mid \;\; &< \;\; \alpha_i \\ \mid (\mu_k)_i \mid \;\; &< \;\; \beta_i \end{aligned}$$

erfüllt sind, dann gilt

$$\frac{\partial}{\partial \varepsilon} T^1(w_k + \varepsilon \Delta w_k) \mid_{\varepsilon=0} < 0 \;\; \Longleftrightarrow \;\; \Delta w_k \neq 0$$

Mit anderen Worten: Falls die *Strafparameter* α, β groß genug sind, bewirkt dies, daß das QP-Update auch die T^1-Funktion verkleinert und umgekehrt.
Man nennt T^1 eine *exakte Penalty-Funktion*, was so definiert ist:

DEFINITION 2.11
T heißt *exakte Penalty-Funktion*, wenn folgende Eigenschaft erfüllt ist:
Sei w^* lokales Minimum des beschränkten nichtlinearen Problems, dann ist w^* auch lokales Minimum des unbeschränkten Problems mit T.

Die T^1-Funktion ist nichtdifferenzierbar, man kann aber mit dem SQP-Verfahren trotzdem Abstiegsrichtungen erhalten, denn T^1 hat nur endlich viele Knicke und ist stetig, sodaß wir in jedem Punkt Richtungsableitungen definieren können:

$$\Theta(w_k,\Delta w_k) \quad := \quad \lim_{\alpha \longrightarrow 0+} \quad \frac{T^1(w_k + \alpha \Delta w_k) - T^1(w_k)}{\alpha}$$

Die Ableitung ist einseitig und nicht notwendig identisch mit der Ableitung auf der anderen Seite.

Es läßt sich zeigen, daß eine Liniensuche bezüglich T^1 tatsächlich zu globaler Konvergenz führt.

SATZ 2.12 (ABSTIEGSRICHTUNGEN BEI DER LINIENSUCHE)
Falls Δw_k die Lösung des $QP(w_k,B_k)$ und B_k positiv definit und $\alpha_i > \mid (\lambda_k)_i \mid$, $\beta_i > \mid (\mu_k)_i \mid$ ist, dann ist $\Theta(w_k, \Delta w_k) < 0$, das heißt Δw_k ist Abstiegsrichrung für T^1.

SATZ 2.13 (GLOBALE KONVERGENZ DES SQP-VERFAHRENS)
Führt man SQP-Verfahren mit exakter Liniensuche für

$$\min_{0 \le \alpha \le 1} \quad T^1(w_k + \alpha \Delta w_k)$$

durch und sind folgende Voraussetzungen erfüllt

- $\alpha_i \ge \mid (\lambda_k)_i \mid, \quad \beta_i \ge \mid (\mu_k)_i \mid$
- $\forall\, \xi \in \mathbb{R}^n \quad \varepsilon \mid \xi \mid^2 \le \xi^T B_k \xi \le C \mid \xi \mid^2$ *mit $\varepsilon > 0$ und $C < \infty$*
- *Die Menge $N(w^0) := \{w : T^1(w) \le T^1(w^0)\}$ ist kompakt (d.h. insbesondere beschränkt)*

dann gilt:
Entweder existiert ein k, sodaß (w_k, λ_k, μ_k) ein KKT-Punkt des ursprünglichen nichtlinearen Problems ist, oder (w_k) hat einen Häufungspunkt in $N(w^0)$ und jeder Häufungspunkt ist ein KKT-Punkt des nichtlinearen Problems.

In der praktischen Realisierung gibt es einige zu beachtende Punkte.

- Man muß die α_i, β_i von vornherein so wählen, daß $\alpha_i \ge \mid (\lambda_k)_i \mid \quad \forall\, k$. Ansonsten würde sich T^1 dauernd ändern. In der Praxis wählt man $(\alpha_k)_i := \max\{\mid (\lambda_k)_i \mid, \frac{1}{2}((\lambda_k)_i + (\alpha_{k-1})_i)\}$. (Wir behalten also bei Verringerung von $\mid (\lambda_k)_i \mid$ etwas Spielraum.)

- Statt einer exakter Liniensuche wird eine Armijo-Goldstein-Strategie gewählt (siehe vorhergehenden Abschnitt):

$$\begin{aligned} w_{k+1} &:= w_k + \alpha_k \Delta w_k \\ \text{wobei } \alpha_k &:= \max_m \quad \{0 < \beta^m < 1 : T^1(w_k + \beta^m \Delta w_k) < T^1(w_k) + \Theta(w_k, \Delta w_k)\beta^m \mu\} \\ &\quad \text{mit } 0 < \mu < \frac{1}{2} \text{ und } 0 < \beta < 1 \end{aligned}$$

  Auch für dieses Verfahren läßt sich globale Konvergenz zeigen.

2.2.4.3 Maratos-Effekt

Durch den Einsatz von bestimmten Schrittweitenstrategien kann im ungünstigsten Fall die superlineare Konvergenz zerstört werden, wenn der sogenannte *Maratos-Effekt* ([Mar78]) auftritt. Sind die Nebenbedingungen nichtlinear, d.h. irgendwie gekrümmt, wird bei der Linearisierung der Nebenbedingungen innerhalb des SQP-Verfahrens diese Krümmung nicht berücksichtigt. Dies führt dazu, daß Vollschritte, die eigentlich einen guten Fortschritt in Richtung Lösung bringen, zurückgewiesen werden, weil die Gütefunktion sich verschlechtert. Das heißt, wir machen immer nur Teilschritte statt Vollschritte, was die Konvergenzgeschwindigkeit beeinträchtigt. Wir wollen im folgenden drei Möglichkeiten vorstellen, wie der Maratos-Effekt vermieden werden kann: die Verwendung von Fletchers erweiterter Lagrange-Funktion, die Second-Order-Correction und die Watchdogtechnik.

Die erste Variante ist der Gebrauch einer alternativen Gütefunktion, nämlich Fletchers erweiterter Lagrange-Funktion (Fletcher's augmented Lagrangian function):

$$T_w = F(w) - \lambda(w)^T G(w) + \frac{1}{2\mu}\|G(w)\|^2$$

Diese Gütefunktion ist differenzierbar und garantiert, daß in der Nähe der Lösung immer Vollschritte gemacht werden. Die Methode ist aber nur bei Problemen ohne Ungleichungsnebenbedingungen anwendbar.

Eine weitere Methode zur Verhinderung des Maratos-Effekt ist die *Second-Order-Correction*. Man löst in jedem Schritt ein zweites quadratisches Problem für ein zweites Update δ_k:

$$\begin{aligned} &\min \quad (\delta_k)^T \delta_k \\ G_i(w_k + y_k) + \nabla G_i(w_k)^T \delta_k &= 0 \\ H_i(w_k + y_k) + \nabla H_i(w_k)^T \delta_k &\geq 0 \end{aligned}$$

Dieses zweite Update ist sozusagen eine Projektion von $w_k + \Delta w_k$ auf den linearisierten zulässigen Bereich. Man bestimmt dann den Punkt w_{k+1} durch Liniensuche entlang der Parabel $w_k + \alpha \Delta w_k + \alpha^2 \delta_k$ (siehe [Fle82]). In der Praxis benutzt man $\nabla G_i(w_k)$ statt $\nabla G_i(w_k + \Delta w_k)$ und $\nabla H_i(w_k)$ statt $\nabla H_i(w_k + \Delta w_k)$, um Rechenaufwand zu sparen.
Folgendes Beispiel veranschaulicht die Second-Order-Correction:

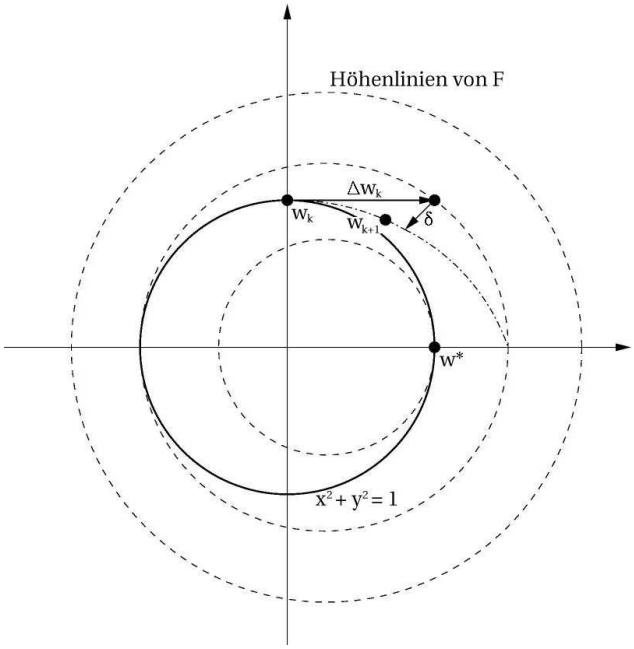

Abbildung 2.1: Beispiel für Second-Order-Correction:
$\min_{x,y} F(x,y) = 2(x^2 + y^2 - 1) - x$
$x^2 + y^2 - 1 = 0$

Sei

$$\begin{aligned} \min_{x,y} F(x,y) &= 2(x^2+y^2-1)-x \\ x^2+y^2-1 &= 0 \end{aligned}$$

Das Beispiel ist dargestellt in Abbildung 2.1. Die optimale Lösung ist $x^* = (1,0)^T$. Sei $x^k = (0,1)^T$; das SQP-Verfahren berechnet den Schritt $\Delta w_k = (1,0)^T$. Obwohl der Schritt gut ist, wird er zurückgewiesen, weil die Gütefunktion T^1 sich verschlechtert. Die Second-Order-Correction berechnet ein zweites Update δ als Projektion auf die Linearisierung von $x^2+y^2-1=0$. Die Liniensuche sucht dann entlang der eingezeichneten Parabel. Neue SQP-Iterierte ist w_{k+1} wie eingezeichnet.

Die *Watchdog-Technik* ([CLPP82]) verwendet zwei verschiedene Kriterien für eine Akzeptanz der Schrittlänge: ein Standard-Kriterium und ein relaxiertes Kriterium (z.B. einfach immer Vollschritte akzeptieren), das weniger restriktiv ist als das erste. Zunächst verwendet man das relaxierte Kriterium und erlaubt dabei, daß in einem einzelnen Schritt die Gütefunktion zunimmt, sofern ein Abstieg nach einer bestimmen Anzahl von t Schritten erreicht wird. Dazu werden mithilfe einer Monitorstrategie die letzten Richtungen und Schritte mitgeführt. Ist einmal auch nach t Schritten kein Abstieg erzielt, springt man auf einen geeigneten früheren Schritt, der noch einen Abstieg erzielte, zurück und führt von dort aus eine oder mehrere Iterationen mit dem Standardkriterium durch, um aus dem kritischen Punkt herauszukommen. Danach versucht man es dann wieder mit dem relaxierten Kriterium.

2.2.4.4 Trust-Region-Technik

Für SQP-Methoden, die Update-Formeln verwenden, die keine positive Definitheit der Approximationen garantieren, oder für SQP-Methoden mit exakten oder durch finite Differenzen berechnete Hesse-Matrizen ist die Liniensuche nicht geeignet, da indefinite projizierte Hesse-Matrizen auftreten können. Sogar wenn unbeschränkte quadratische Programme durch Schranken an alle Variablen ausgeschlossen werden, kann die Globalisierungsstrategie scheitern, da viel zu große Schritte gemacht werden. Eine sinnvolle Strategie ist daher die Beschränkung der Schrittlänge $\|\Delta w\| \leq \rho_k$. Damit wird sowohl das Auftreten von indefiniten projizierten Hesse-Matrizen verhindert als auch sichergestellt, daß die neue Iterierte $w_k + \Delta w_k$ in einem *Vertrauensgebiet* um w_k bleibt, in dem das quadratische Problem eine gute Näherung des ursprünglichen NLPs ist. Das ist die Grundidee der *Trust-Region-Technik*. Entscheidend für eine gute Trust-Region-Technik ist die adaptive Steuerung des *Trust Radius*.

Wir arbeiten mithilfe einer Gütefunktion, die eine Modifikation der Standardgütefunktion ist:

$$\hat{T}^1(w) := \frac{1}{\xi} f(w) + \sum_{i=1}^{l} \alpha_i \mid g_i(w) \mid + \sum_{i=1}^{m} \beta_i \mid \min(0, h_i(w)) \mid$$

mit einem zusätzlichen Gewichtungsfaktor $\xi \geq 1$, der dazu dient, daß auch bei indefiniten Approximationen der Hesse-Matrix Kompatibilität zwischen Suchrichtung und Gütefunktion garantiert ist. Mithilfe dieser Gütefunktion läßt sich eine Bedingung formulieren (siehe [Lei99]), die festlegt, ob ein Schritt akzeptiert wird oder nicht. Wird der Schritt akzeptiert und ist die Beschränkung $\|\Delta w\| \leq \rho_k$ aktiv, wird der Trust-Radius für das aktuelle QP vergrößert (z.B. verdoppelt) und das aktuelle QP wird mit der modifizierten Beschränkung gelöst. Wird der Schritt nicht akzeptiert, wird der Trust-Radius für das aktuelle QP verkleinert (z.B. halbiert) und das aktuelle QP wird mit der modifizierten Beschränkung gelöst.
Eine gute Erweiterung der Trust-Region-Technik ist die Verwendung einer Second-Order-Correction. Es hat sich gezeigt, daß so der Trust-Region-Radius oft in sinnvoller Weise vergrößert werden kann und die Anzahl der SQP-Iterationen dadurch stark sinkt.

2.2.4.5 Filter-Methoden

Die Idee der *Filter-Methoden* ist, daß eine neue Iterierte akzeptiert wird, wenn *entweder* die Zielfunktion verbessert wird *oder* die Zulässigkeit verbessert wird anstatt eine Kombination der beiden, die durch eine Gütefunktion definiert wird. Diese Idee wurde erstmals von Fletcher und Leyffer ([FL02]) sowie von Wächter und Biegler ([WB02]) entwickelt. Mit Wertepaaren für Zielfunktionswert und einem Maß für die Zulässigkeit aus bestimmten vergangener SQP-Iterierten wird ein Filter gebildet, der garantiert, daß Kreiseln, sogenanntes Cycling verhindert wird. Dieser Filter kann so gestaltet werden, daß die Akzeptanz einer neuen SQP-Iterierten bezüglich des Filters weniger restringent ist als bei der Reduktion einer Niveaufunktion. Falls die neue Iterierte nicht vom Filter akzeptiert wird, wird der Schrittweitenparameter mittels einer Liniensuche solange reduziert, bis dies erfüllt ist.
Im Gegensatz zur Globalisierung mit der Liniensuche und geeigneter Gütefunktion ist die Filter-Methode nicht skalierungsinvariant bezüglich Variablen und Funktionen.

2.2.5 Reduzierte Verfahren

Ist die Zahl der Freiheitsgrade eines Systems deutlich geringer als die Zahl der Variablen, ist der Einsatz von reduzierten Verfahren sinnvoll. Bei den reduzierten SQP-Verfahren (RSQP-Verfahren) benutzt man eine Zerlegung des Suchraums, sodaß die eigentliche Optimierung (Gradientenberechnung und Approximation der Hesse-Matrix) nur auf einem Tangentialraum der Nebenbedingungen stattfindet. Seit ca. 1980 wurden verschiedene Varianten von reduzierten SQP-Methoden entwickelt – siehe zum Beispiel [Gab82]. Neuere Entwicklungen auf diesem Gebiet sind nachzulesen unter [BNS95], [Sch96] und [Sch04].

Wir betrachten zunächst das gleichungsbeschränkte Optimierungsproblem:

$$\min_{w_1,w_2} \quad F(w_1,w_2) \tag{2.6a}$$

$$G(w_1,w_2) \quad = \quad 0 \tag{2.6b}$$

wobei $w_1 \in \mathbb{R}^{n_1}$ und $w_2 \in \mathbb{R}^{n_2}$, $G : \mathbb{R}^{n_1+n_2} \rightarrow \mathbb{R}^{n_1}$, F,$G \in \mathcal{C}^2$ und $\nabla_{w_1} G$ nichtsingulär.

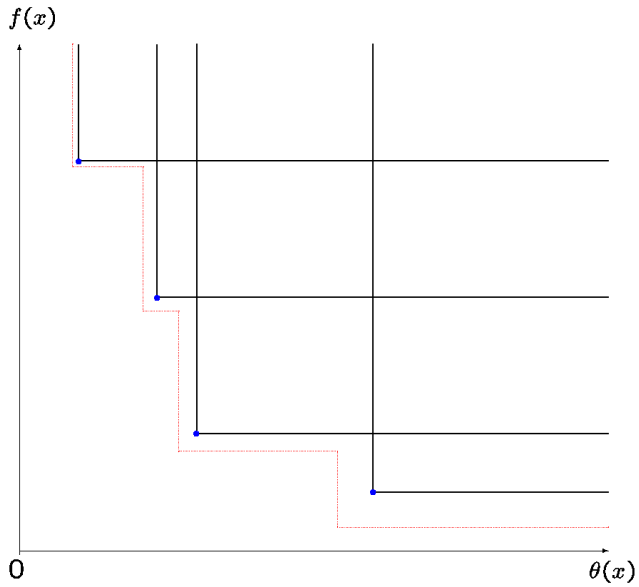

Abbildung 2.2: Filter-Methode
$f(x)$ Zielfunktion, $\theta(x)$ Maß für Zulässigkeit, gestrichelt: Filter

Eine neue Iterierte wird akzeptiert, wenn entweder die Zielfunktion verbessert wird oder die Zulässigkeit verbessert wird

Eine mögliche Methode, das obige Optimierungsproblem zu lösen, ist w_1 als Funktion von w_2 zu betrachten und so ein unbeschränktes Optimierungsproblem nur für w_2 zu erhalten. Man wendet also den Satz über implizite Funktionen an, der garantiert, daß es eine Funktion $\vartheta : \Omega \subseteq \mathbb{R}^{n_2} \rightarrow \mathbb{R}^{n_1}$ gibt mit $w_1 = \vartheta(w_2) \forall w_1 \in \Omega$. Dann läßt sich das beschränkte Optimierungsproblem (2.6) überführen in ein unbeschränktes:

$$\min_{w_2} \quad \tilde{F}(w_2) \quad := \quad F(\vartheta(w_2),w_2) \tag{2.7}$$

Methoden, die auf dieser simplen Reduktion der Variablen basieren, heißen *Feasible-Path-Methoden*.
Die Vorteile dieser Vorgehensweise sind die reduzierte Anzahl der Optimierungsvariablen und die Tatsache, daß die Hesse-Matrix von $\tilde{F}(w_2)$ positiv semi-definit in der Lösung ist, wenn es eine Lösung gibt. Der Nachteil der Methode ist, daß in jedem Schritt eines Lösungsalgorithmus für Problem (2.7) die Gleichung $G(w_1,w_2) = 0$ gelöst werden muß, was in den meisten Fällen wiederum iterativ geschehen muß. Außerdem ist das Problem (2.7) typischerweise um einiges nichtlinearer als das Problem (2.6).
Der numerische Aufwand, um einen zulässigen Punkt zu erhalten, ist sehr groß. Solange w_2 nicht optimal ist, ist es jedoch auch nicht nötig, daß w_1 sehr genau ist. Warum soll man also nicht auf eine exakte Berechnung von w_1 verzichten und lediglich einen Newtonschritt "zurück" zur Mannigfaltigkeit $G(w_1,w_2) = 0$ durchführen? Dies führt zu den reduzierten SQP-Verfahren.

Wir betrachten das gleichungsbeschränkte NLP:

$$\begin{aligned} \min F(w) \\ G(w) \quad = \quad 0 \end{aligned}$$

mit $F : \mathbb{R}^n \rightarrow \mathbb{R}$ und $G : \mathbb{R}^n \rightarrow \mathbb{R}^l$. Ein gewöhnliches SQP-Verfahren löst in jedem Schritt folgendes quadratische Programm:

$$\begin{aligned} \min_{\Delta w} \nabla F_k^T \Delta w + \tfrac{1}{2} \Delta w^T \nabla_w^2 \mathcal{L}_k \Delta w \\ G_k + \nabla G_k^T \Delta w \quad = \quad 0 \end{aligned}$$

wobei für die Hesse-Matrix $\nabla_w^2 \mathcal{L}_k$ eine Approximation verwendet wird. Um eine geeignete Zerlegung des Suchraums zu erhalten, wählen wir eine nicht-singuläre Matrix

$$S_k \quad = \quad \begin{pmatrix} S_k^{\mathcal{R}} & S_k^{\mathcal{N}} \end{pmatrix} \qquad \text{mit } S_k^{\mathcal{R}} \in \mathbb{R}^{n \times m}, S_k^{\mathcal{N}} \in \mathbb{R}^{n \times (n-m)}$$

sodaß $S_k^{\mathcal{N}}$ eine Basis des Nullraums der Jacobi-Matrix der Nebenbedingungen ist, also:

$$\nabla G_k^T S_k^{\mathcal{N}} \quad = \quad 0$$

Nun läßt sich der Schritt schreiben als:

$$\Delta w_k \quad = \quad S_k^{\mathcal{R}} y_k^{\mathcal{R}} + S_k^{\mathcal{N}} y_k^{\mathcal{N}} \tag{2.8}$$

Es läßt sich leicht zeigen, daß $y_k^{\mathcal{R}}$ durch

$$(\nabla G_k^T S_K^{\mathcal{R}}) y_k^{\mathcal{R}} \quad = \quad -G_k$$

bestimmt ist und daß $y_k^{\mathcal{N}}$ das folgende Optimierungsproblem löst:

$$\min_{y^{\mathcal{N}}} \left((S_k^{\mathcal{N}})^T \nabla F_k + (S_k^{\mathcal{N}})^T \nabla_w^2 \mathcal{L}_k S_k^{\mathcal{R}} y_k^{\mathcal{R}} \right)^T y^{\mathcal{N}} + \tfrac{1}{2} (y^{\mathcal{N}})^T \left((S_k^{\mathcal{N}})^T \nabla_w^2 \mathcal{L}_k S_k^{\mathcal{N}} \right) y^{\mathcal{N}} \qquad (2.9)$$

Die meisten RSQP-Verfahren vernachlässigen den Term $(S_k^{\mathcal{N}})^T \nabla_w^2 \mathcal{L}_k S_k^{\mathcal{R}} y_k^{\mathcal{R}}$ in (2.9) und verwenden eine direkte Update-Strategie für die reduzierte Hesse-Matrix $(S_k^{\mathcal{N}})^T \nabla_w^2 \mathcal{L}_k S_k^{\mathcal{N}}$. Der Nullraumschritt berechnet sich dann also aus:

$$\min_{y^{\mathcal{N}}} \left((S_k^{\mathcal{N}})^T \nabla F_k \right)^T y^{\mathcal{N}} + \tfrac{1}{2} (y^{\mathcal{N}})^T B_k^{\mathcal{N}} y^{\mathcal{N}}$$

wobei $B_k^{\mathcal{N}}$ eine Approximation der reduzierten Hesse-Matrix ist. Solche RSQP-Verfahren konvergieren 2-Schritt-superlinear (siehe [NO85]).
Eine natürliche Wahl für die Zerlegung des Suchraums ist:

$$\begin{aligned} S_k^{\mathcal{N}} &= \begin{pmatrix} -(\nabla_{w_1} G_k^T)^{-1} \nabla_{w_2} G_k^T \\ I \end{pmatrix} \\ S_k^{\mathcal{R}} &= \begin{pmatrix} I \\ 0 \end{pmatrix} \end{aligned}$$

Diese sogenannte *Koordinatenbasis (coordinate base approach bzw. separability framework)* erlaubt effiziente Implementierungen, da die dünnbesetzte Struktur von $\nabla w_1 G_k$ ausgenutzt werden kann.
Eine weitere gebräuchliche Wahl für diese Operatoren mit orthogonalen Basen ergibt sich aus der QR-Zerlegung von $\nabla_w G_k$. Die aus diesen *orthogonalen Basen (orthogonal framework)* resultierenden reduzierten SQP-Methoden heißen *projizierte SQP-Verfahren*. Sie führen zu gut konditionierten Teilproblemen und numerisch stabiler Elimination. Aber sie haben auch Nachteile: Um die Konvergenz von Algorithmus 2.2.5 zu garantieren, müssen die Operatoren $S_k^{\mathcal{N}}$ und $S_k^{\mathcal{R}}$ lipschitz-stetig sein in Bezug auf w. Das kann bei einer normalen Householder-Zerlegung von $\nabla_w G_k$ nicht in jeder Iteration gewährleistet werden. Spezielle Rechenschemata zur Berechnung einer Orthonormal-Basis $S_k^{\mathcal{N}}$ müssen eingeführt werden, die auf QR-Zerlegungs-Techniken beruhen und garantieren, daß $S_k^{\mathcal{N}}$ stetig bleibt. Solche Zerlegungen der Matrix $\nabla_w G_k$ werden sehr aufwendig, wenn die Dimension des Problems größer wird.
Die Matrizen $S_k^{\mathcal{N}}$ und $S_k^{\mathcal{R}}$ müssen nicht vollständig gespeichert werden. Die einzige Form, in der sie gebraucht werden, ist als Matrix-Vektor-Multiplikation.

2.2.6 Abbruchbedingung

Eine geeignetes Kriterium, das überprüft, ob Konvergenz zu einem lokalen Optimum (mit einer akzeptablen Genaugkeit) erreicht wurde, ist nötig, um als Abbruchbedingung für den SQP-Algorithmus zu dienen. Gebräuchlich ist die Verwendung des Lagrange-Gradienten:

Falls x^k zulässig ist und

$$\|\nabla \mathcal{L}(x^k,\lambda^k,\mu^k)\| \quad < \quad \varepsilon_{Tol}$$

dann stoppe den Algorithmus mit w^k,λ^k,μ^k als Lösung. (ε_{tol} ist dabei eine vorgegebene Toleranz für die Abbruchbedingung, z.B. $\varepsilon_{tol} = 10^{-4}$)

Wir verwenden eine andere, ebenfalls verbreitete Abbruchbedingung, die eine gewichtete Summe der möglichen Verbesserungen im Zielfunktional und der Verletzungen der Nebenbedingungen betrachtet:

$$\|\nabla F(w_k)^T \Delta w_k\| + \sum_{i=1}^{l} \mid (\tilde{\lambda}_k)_i G_i(w_k) \mid + \sum_{i=1}^{m} \mid (\tilde{\mu}_k)_i H_i(w_k) \mid \quad \leq \quad \varepsilon_{Tol}$$

2.2.7 SQP-Algorithmus

In diesem Abschnitt ist ein vollständiger SQP-Algorithmus aufgeführt.

ALGORITHMUS 2.3 (SQP-VERFAHREN)

1. Beginne mit $k := 0$ und einem Startwert w_0
2. Berechne die Zielfunktion, Nebenbedingungen und alle Ableitungen: $F(w_0)$, $G(w_0)$, $H(w_0)$, $\nabla F(w_0)$, $\nabla G(w_0)$, $\nabla H(w_0)$
3. Wähle eine Anfangsnäherung für die Hesse-Matrix B_0
4. Löse das aktuelle quadratische Programm

$$\begin{aligned} \min_{\Delta w_k} \frac{1}{2}\Delta w_k^T B_k \Delta w + \nabla F(w_k)^T \Delta w_k \\ G(w_k) + \nabla G(w_k)\Delta w_k &= 0 \\ H(w_k) + \nabla H(w_k)\Delta w_k &\geq 0 \end{aligned}$$

   und erhalte Δw_k sowie die Lagrange-Multiplikatoren $\tilde{\lambda}_k, \tilde{\mu}_k$
5. Konvergenzcheck: Falls

$$\|\nabla F(w_k)^T \Delta w_k\| + \sum_{i=1}^{l} \mid (\tilde{\lambda}_k)_i G_i(w_k) \mid + \sum_{i=1}^{m} \mid (\tilde{\mu}_k)_i H_i(w_k) \mid \quad \leq \quad \varepsilon_{Tol}$$

   erfüllt ist, beende Algorithmus mit $w_k, \tilde{\lambda}_k, \tilde{\mu}_k$ als Lösung.
6. Führe eine Liniensuche durch und erhalte t_k aus einer Näherungslösung von

$$\min_{t_k} T(w_k + t_k \Delta w_k)$$

   mit geeigneter Gütefunktion T.
7. Führe die Iteration durch

$$\begin{aligned} w_{k+1} &:= w_k + t_k \Delta w_k \\ \lambda_{k+1} &:= \lambda_k + t_k(\tilde{\lambda}_k - \lambda_k) \text{ (oder } \lambda_{k+1} := \tilde{\lambda}_k) \\ \mu_{k+1} &:= \mu_k + t_k(\tilde{\mu}_k - \mu_k) \text{(oder } \mu_{k+1} := \tilde{\mu}_k) \end{aligned}$$

8. Berechne "alten" Gradienten mit neuen Multiplikatoren

$$\nabla_w \mathcal{L}(w_k, \lambda_{k+1}, \mu_{k+1}) \quad = \quad \nabla F(w_k) - \nabla G(w_k)\lambda_{k+1} - \nabla H(w_k)\mu_{k+1}$$

   (Leicht zu berechnen)

9. Berechne Funktionen und Ableitungen am neuen Punkt w_{k+1}:
   $F(w_{k+1})$, $G(w_{k+1})$, $H(w_{k+1})$, $\nabla F(w_{k+1})$, $\nabla G(w_{k+1})$, $\nabla H(w_{k+1})$
10. Berechne "neuen" Gradienten

$$\nabla_w \mathcal{L}(w_{k+1},\lambda_{k+1},\mu_{k+1}) \;=\; \nabla F(w_{k+1}) - \nabla G(w_{k+1})\lambda_{k+1} - \nabla H(w_{k+1})\mu_{k+1}$$

11. Berechne Update der Hesse-Matrix

$$B_{k+1} \;=\; B_k + U(B_k,\delta_k,\gamma_k)$$

    aus

$$\begin{aligned} p_k &= w_{k+1} - w_k \\ q_k &= \nabla_w \mathcal{L}(w_{k+1},\lambda_{k+1},\mu_{k+1}) - \nabla_w \mathcal{L}(w_k,\lambda_{k+1},\mu_{k+1}) \end{aligned}$$

    Dabei bezeiche $U(B_k, p, q)$ eine geeignete Update-Formel.
12. $k := k + 1$ und gehe zu Schritt 3.

2.3 SQP-Verfahren für Probleme der optimalen Steuerung

In den folgenden Abschnitten werden die Besonderheiten unseres SQP-Verfahrens erläutern, das speziell auf Probleme der optimalen Steuerung zugeschnitten ist. Dabei werden wir eingehen auf die effiziente Berechnung von Ableitungen durch interne numerische Differentiation, die Struktur des quadratischen Programms, spezielle Update-Techniken für blockdiagonale Hesse-Matrizen, einem Kondensierungs-Algorithmus bei der Lösung des quadratischen Programms und an die Problemstruktur angepaßte partiell reduzierte Verfahren.

2.3.1 Interne numerische Differentiation

Die meisten Funktionen und Ableitungen, die zum Aufstellen des quadratischen Programms notwendig sind, sind direkt duch explizite Formeln gegeben: Φ_i, g_i, h_i, c_i, r_i^s, r_i^e aus (1.11). Die entsprechenden Ableitungen können über finite Differenzen oder besser – da genauer und effizienter – durch automatische Differentiation berechnet werden. Die Zustandsvariablen $x(t_{i,j+1}; s_{ij}^x, s_{ij}^z, q_{ij}, p, v)$, die in die Stetigkeitsbedingungen eingehen, werden dagegen durch eine numerische Integration eines Anfangswertproblems berechnet. Um die entsprechenden Jacobi-Matrizen berechnen zu können, müssen die Variations-Trajektorien

$$\frac{\partial x(t; s_{ij}^x, s_{ij}^z, q_{ij}, p, v)}{\partial s_{ij}^x}, \frac{\partial x(t; s_{ij}^x, s_{ij}^z, q_{ij}, p, v)}{\partial s_{ij}^z},$$
$$\frac{\partial x(t; s_{ij}^x, s_{ij}^z, q_{ij}, p, v)}{\partial q_{ij}}, \frac{\partial x(t; s_{ij}^x, s_{ij}^z, q_{ij}, p, v)}{\partial p}, \frac{\partial x(t; s_{ij}^x, s_{ij}^z, q_{ij}, p, v)}{\partial v}$$

zusammen mit der Lösung $x(t; s_{ij}^x, s_{ij}^z, q_{ij}, p, v)$ berechnet werden. Da die Ableitungen mit einer gewissen für die Optimierung nötigen Genauigkeit benötigt werden, kostet die Berechnung dieser Ableitungen typischerweise den größten Anteil an der Gesamtrechenzeit.

Die einfachste Methode, die nötigen Ableitungen der Zustandsvariablen nach Anfangswerten, Steuergrößen und Parametern zu berechnen, ist die erneute Lösung des Anfangswertproblems für leicht gestörte Systeme (variierte Trajektorien). Man erhält durch finite Differenzen eine Approximation für die Ableitungen von erster Ordnung (bzw. zweiter Ordnung bei zentralen Differenzenquotienten). Betrachtet man dabei den Integrationsprozeß als "Black Box", nennt man diese Methode gewöhnlich *externe numerische Differentiation*, da die Differentiation der Trajektorie außerhalb des Integrators geschieht. Sie birgt jedoch einige Nachteile. Wie man weiß, ist die Ausgabe eines modernen, adaptiven Integrators im allgemeinen eine nicht-stetige Funktion der Anfangswerte, Steuergrößen und Parameter. Schon für leicht gestörte Anfangswerte, Steuerungen oder Parameter können sich Entscheidungen über Schrittweiten (z.B. Schrittzurückweisungen) oder Ordnungssteuerungen ändern, das heißt der Algorithmus wird veranlaßt, einen anderen Pfad bei der Durchführung der Einzelschritte zu gehen. Aufgrund dieser diskreten Entscheidungen sollte klar sein, daß bei einer Verwendung des Integrators als "Black Box" durch finite Differentiation keine verläßlichen Ableitungen erhalten werden können. Integratoren ohne adaptive Elemente, das heißt mit fixer Ordnung und Schrittweite, die dieses Problem umgehen würden, sind wesentlich ineffizienter. Die Integrationsgenauigkeit müßte auf ein extrem hohes Level gesetzt werden. Aber selbst im besten Fall kann die Ableitung der Trajektorie nur mit einer Genauigkeit $\varepsilon_A = \sqrt{\varepsilon_T}$ berechnet werden, wenn ε_T die Genauigkeit der Nominaltrajektorie ist.

Abhilfe schafft eine wesentlich effizientere Variante, die 1981 von Bock ([Boc81]) vorgestellt wurde, die *interne numerische Differentiation*. Hier wird bei der Berechnung der gestörten Trajektorien ein identisches Berechnungsschema verwendet wie bei der Berechnung der Nominaltrajektorie.
Die einfachste Version der internen numerischen Differentiation berechnet die variierten Trajektorien mit den gleichen "eingefrorenen" adaptiven Komponenten. Im Gegensatz zur externen

numerischen Differentiation ist es möglich, die adaptiven Komponenten des Integrators zu variieren, solange sie identisch sind für Nominaltrajektorie und variierte Trajektorien. Es bleiben die Nachteile der Methode durch die Schwierigkeit einer geeigneten a-priori-Wahl der Schrittweite ε zur Berechnung der finiten Differenzen und die Ungenauigkeit aufgrund der Approximation erster (bzw. zweiter) Ordnung.
Eine noch effizientere Variante ergibt sich, wenn die Differentiation innerhalb des Diskretisierungsschemas des Integrators geschieht. Die interne numerische Differentiation ist dann stabil auch für niedrige Integrationsgenauigkeiten, da sie eine "exakte" Ableitung des gewählten adaptiven Diskretisierungsschemas berechnet. Diese Methode läßt sich für viele Integrations-Verfahren verwenden (z.B. Einschritt-Verfahren, Mehrschritt-Verfahren und Extrapolations-Verfahren).

Um die Gleichungen für die Berechnung der Ableitungen herzuleiten, gibt es zwei Möglichkeiten: Die erste Möglichkeit ist es, das Diskretisierungsschema zur Lösung des Anfangswertproblems abzuleiten, um so ein Diskretisierungsschema zur Berechnung der Ableitungen zu erhalten. Die zweite Möglichkeit ist es, die Variationsdifferentialgleichung zu benutzen. Diskretisiert man die Variationsdifferentialgleichung mit dem gleichen Diskretisierungsschema wie die ursprüngliche DAE, erhält man dieselben Gleichungen wie bei der ersten Methode, bei der einfach das Diskretisierungsschema zur Lösung des Anfangswertproblems abgeleitet wird. Diese Kommutativität von Integrationsschema und Differenzierung führt zu exakten numerischen Ableitungen der numerischen Lösung der DAE, wenn exakte Ableitungen der Modellgleichungen f und g (z.B. bei automatischer Differentiation) zur Verfügung stehen. Bock ([Boc83]) bezeichnet dies als *analytischen Grenzfall (analytical limit)* der internen numerischen Differentiation.

Die Variationsdifferentialgleichung für die Mehrziel-Methode erhält man, wenn man die DAE auf einem Mehrzielintervall 1.8 nach $(s_i^x, s_i^z, q, p)^T$ differenziert:

$$
\begin{aligned}
A(\cdot) \begin{pmatrix} \dot{\mathcal{W}}^x_{s_i^x} \\ \dot{\mathcal{W}}^x_{s_i^z} \\ \dot{\mathcal{W}}^x_q \\ \dot{\mathcal{W}}^x_p \end{pmatrix} &= \begin{pmatrix} -A_x\dot{x} + f_x \\ -A_z\dot{x} + f_z \\ -A_q\dot{x} + f_q \\ -A_p\dot{x} + f_p \end{pmatrix}^T \begin{pmatrix} \mathcal{W}^x_{s_i^x} & \mathcal{W}^x_{s_i^z} & \mathcal{W}^x_q & \mathcal{W}^x_p \\ \mathcal{W}^z_{s_i^x} & \mathcal{W}^z_{s_i^z} & \mathcal{W}^z_q & \mathcal{W}^z_p \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{pmatrix} \\
0 &= \begin{pmatrix} g_x \\ g_z \\ g_q \\ g_p \end{pmatrix}^T \begin{pmatrix} \mathcal{W}^x_{s_i^x} & \mathcal{W}^x_{s_i^z} & \mathcal{W}^x_q & \mathcal{W}^x_p \\ \mathcal{W}^z_{s_i^x} & \mathcal{W}^z_{s_i^z} & \mathcal{W}^z_q & \mathcal{W}^z_p \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{pmatrix} - \alpha(t) \begin{pmatrix} g_x(\tau_j) \\ g_q(\tau_j) \\ g_p(\tau_j) \end{pmatrix}^T
\end{aligned}
$$

mit den Anfangswerten

$$
\begin{pmatrix} \mathcal{W}^x_{s_i^x} \\ \mathcal{W}^x_{s_i^z} \\ \mathcal{W}^x_q \\ \mathcal{W}^x_p \end{pmatrix}(t_i) = \begin{pmatrix} I \\ I \\ 0 \\ 0 \end{pmatrix}
$$

wobei $\mathcal{W}$ die Wronski-Matrizen der Trajektorien bezeichnet, also zum Beispiel:

$$\mathcal{W}^x_{s_i^x} = \frac{\partial x}{\partial s_i^x}(t; s_i^x, s_i^z, q, p)$$

Es gibt zwei Möglichkeiten, um Lösungen der VDAE zu erhalten. Wendet man die Diskretisierung der DAE auch auf die VDAE an, erhält man ein lineares System, das sich direkt lösen läßt. Diese erste Variante ist sinnvoll bzw. effizient, wenn im Verhältnis zur Zahl der Zustände viele Ableitungen berechnet werden müssen Die zweite Möglichkeit der Berechnung ergibt sich aus dem Newton-Verfahren zur Lösung des bei der Diskretisierung der DAE entstandenen nichtlinearen Gleichungssystems. Differenziert man das Newton-Verfahren erhält man unter Anwendung der Kettenregel eine Iterationsvorschrift für die Wronski-Matrizen. Die zweite Variante ist sinnvoll bei einer großen Anzahl von Zuständen und einer kleiner Anzahl von Anfangswerten, Parametern und Steuergrößen. (Siehe [Bau99])

Um die Ableitungen gemäß dem hergeleiteten Schema zu berechnen, behandeln wir die Variationsdifferentialgleichung zusammen mit der ursprünglichen DAE. Die Integration geschieht gleichzeitig, Funktionswerte und Ableitungen werden simultan mit der gleichen Folge von Ordnungen und Schrittweiten berechnet. Die Wiederverwendung der Lineare-Algebra-Komponenten macht die interne numerische Differentiation dabei äußerst effizient.

Weitere Erläuterungen und Anwendungen der internen numerische Differentiation finden sich zum Beispiel in [Bau99] und [Sch04].

2.3.2 Struktur des quadratischen Programms

Zur Formulierung eines SQP-Algorithmus' für das in Abschnitt 1.3.5 beschriebene Problem der optimalen Steuerung verwenden wir folgende Anordnung der Variablen:

$$w = \begin{pmatrix} w_{0,0} \\ w_{0,1} \\ \vdots \\ w_{M-1,\hat{m}} \end{pmatrix} \quad \text{mit } w_{ij} := \begin{pmatrix} s_{ij}^x \\ s_{ij}^z \\ \bar{q}_{ij} \end{pmatrix}, \bar{q}_{ij} := \begin{pmatrix} p_{ij} \\ v_{ij} \\ q_{ij} \end{pmatrix}, \hat{m} := m_{M-1}$$

Man beachte, daß am Endpunkt des letzten Mehrzielintervalls in der letzten Stufe keine Steuerungen definiert sind.
Dann ergeben sich für das quadratische Programm

$$\begin{aligned} \min_{\Delta w \in \Omega_k} \tfrac{1}{2} \Delta w^T B_k \Delta w + \nabla_w F(w_k)^T \Delta w \\ G(w_k) + \nabla_w G(w_k)^T \Delta w &= 0 \\ H(w_k) + \nabla_w H(w_k)^T \Delta w &\geq 0 \end{aligned}$$

bei Auslassung des Iterationsindex k folgende Strukturen:

Die Jacobi-Matrix der Nebenbedingungen ist von spezieller Gestalt. Nimmt man alle Gleichungen und Ungleichungen zusammen, ergibt sich folgende Matrix:

$$\begin{pmatrix} X \\ V \\ R \end{pmatrix}$$

Dabei ist X die Jacobi-Matrix der Stetigkeitsbedingungen (1.11b) und Stufenübergangsbedingungen (1.11e):

$$X \;=\; \begin{pmatrix} X^x_{0,0} & X^z_{0,0} & X^{\bar{q}}_{0,0} & -I & & & & & & & & \\ & & & X^x_{0,1} & X^z_{0,1} & X^{\bar{q}}_{0,1} & -I & & & & & \\ & & & & & & \ddots & & & & & \\ & & & & & & & X^x_{M-1,\hat{m}-1} & X^z_{M-1,\hat{m}-1} & X^{\bar{q}}_{M-1,\hat{m}-1} & -I & 0 \quad 0 \end{pmatrix} \tag{2.10}$$

Hierbei sind in den entsprechenden Zeilen die Ableitungen der Stufenübergangsbedingungen einzufügen; sie haben die gleiche Struktur und enthalten die Blöcke:

$$\begin{pmatrix} C^x_{i,m_i} & C^z_{i,m_i} & C^{\bar{q}}_{i,m_i} & -I \end{pmatrix}$$

V ist die Jacobi-Matrix der Konsistenzbedingungen (1.11c) und der diskretisierten Steuerungs- und Zustandsbeschränkungen(1.11d):

$$V \;=\; \begin{pmatrix} V^x_{0,0} & V^z_{0,0} & V^{\bar{q}}_{0,0} & & & & & & \\ & & & V^x_{0,1} & V^z_{0,1} & V^{\bar{q}}_{0,1} & & & \\ & & & & & & \ddots & & \\ & & & & & & & V^x_{M-1,\hat{m}} & V^z_{M-1,\hat{m}} & V^{\bar{q}}_{M-1,\hat{m}} \end{pmatrix} \tag{2.11}$$

Und R ist die Jacobi-Matrix der linear gekoppelten Mehrpunktbedingungen (1.11f):

$$R \;=\; \begin{pmatrix} R^x_{0,0} & R^z_{0,0} & R^{\bar{q}}_{0,0} & \cdots & & & \cdots & R^x_{M-1,\hat{m}} & R^z_{M-1,\hat{m}} & R^{\bar{q}}_{M-1,\hat{m}} \end{pmatrix} \tag{2.12}$$

Zusammengesetzt ergibt sich die Jacobi-Matrix von $Q(w_k,B_k)$ wie in Abbildung 2.3 abgebildet. Nicht dargestellt sind die Kopplungsbedingungen für die Parameter. Auf sie wird in Abschnitt 2.3.4 näher eingegangen.

Alle Problemfunktionen F, G_i und H_j sind *partiell separierbar*, was bedeutet, daß sie sich als Summe von skalaren Funktionen darstellen lassen, die von disjunkten Untermengen von Variablen abhängt. Insbesondere ist die Lagrange-Funktion partiell separierbar:

$$\mathcal{L}(w,\lambda,\mu) \;=\; \sum_{i=0}^{M-1}\sum_{j=0}^{m_i-1} \mathcal{L}_{ij}(w_{ij},\lambda,\mu) + \mathcal{L}_{M-1,\hat{m}}(w_{M-1,\hat{m}},\lambda,\mu)$$

Die Hesse-Matrix hat daher blockdiagonale Gestalt:

$$\nabla_w^2\mathcal{L}(w,\lambda,\mu) \;=\; \begin{pmatrix} \mathcal{B}_{0,0} & & & \\ & \mathcal{B}_{0,1} & & \\ & & \ddots & \\ & & & \mathcal{B}_{M-1,\hat{m}} \end{pmatrix} \qquad \text{mit } \mathcal{B}_{ij} := \nabla_{w_{ij}}^2\mathcal{L}_{ij}(w_{ij},\lambda,\mu)$$

Die Diagonalstruktur kann ausgenutzt werden durch die Verwendung von Block-Update-Formeln (siehe Abschnitt 2.3.3).

$$
\begin{pmatrix}
X^x_{0,0} & X^z_{0,0} & X^{\bar{q}}_{0,0} & -I & & & & & & & & & \\
 & & & X^x_{0,1} & X^z_{0,1} & X^{\bar{q}}_{0,1} & -I & & & & & & \\
 & & & & & & \ddots & & & & & & \\
 & & & & & & & X^x_{M-1,\hat{m}-1} & X^z_{M-1,\hat{m}-1} & X^{\bar{q}}_{M-1,\hat{m}-1} & -I & & \\
V^x_{0,0} & V^z_{0,0} & V^{\bar{q}}_{0,0} & & & & & & & & & & \\
 & & & V^x_{0,1} & V^z_{0,1} & V^{\bar{q}}_{0,1} & & & & & & & \\
 & & & & & & & & \ddots & & & & \\
 & & & & & & & & & & V^x_{M-1,\hat{m}} & V^z_{M-1,\hat{m}} & V^{\bar{q}}_{M-1,\hat{m}} \\
R^x_{0,0} & R^z_{0,0} & R^{\bar{q}}_{0,0} & \cdots & & & & & & \cdots & R^x_{M-1,\hat{m}} & R^z_{M-1,\hat{m}} & R^{\bar{q}}_{M-1,\hat{m}}
\end{pmatrix}
$$

Abbildung 2.3: Struktur der Jacobi-Matrix von $Q(w_k, B_k)$

2.3.3 Block-Update-Formeln

Jede Anwendung eines Standard-Rang-2-Updates würde die bekannten Blockstrukturen in der Hesse-Matrix sofort zerstören. Eine spezielle Update-Methode, die jeden Diagonalblock von B_k separat bearbeitet, wurde von Bock und Plitt vorgeschlagen ([Pli81],[BP84]).
Wenn die Standard-Rang-2-Update-Formel gegeben ist durch

$$\begin{aligned} B_{k+1} &= B_k + U(B_k,p,q) \\ \text{mit} \quad q &= \nabla_w \mathcal{L}(w_{k+1},\lambda_{k+1},\mu_{k+1},\cdot) - \nabla_w \mathcal{L}(w_k,\lambda_{k+1},\mu_{k+1},\cdot) \\ p &= t_k \Delta w_k \end{aligned}$$

läßt sich eine blockweise Update-Formel schreiben durch:

$$B_{k+1} = B_k + \sum_{i=0}^{M-1} \sum_{j=0}^{\hat{m}_i-1} U_{ij}(B_k,p,q) + U_{M-1,\hat{m}}(B_k,p,q)$$

mit

$$\begin{aligned} U_{ij}(B_k,\delta_k,\gamma_k) &:= U(P_{ij}B_kP_{ij},P_{ij}\delta_k,P_{ij}\gamma_k) \\ P_{ij} &:= \begin{pmatrix} 0 & & & & \\ & \ddots & & & \\ & & I_{ij} & & \\ & & & \ddots & \\ & & & & 0 \end{pmatrix} \end{aligned}$$

In der Praxis werden natürlich nur die Diagonalblöcke $(B_{ij})_k$ von B_k gespeichert und man berechnet direkt

$$(B_{ij})_{k+1} = (B_{ij})_k + U((B_{ij})_k,p_{ij},q_{ij}) \tag{2.13}$$

Wie sich in ausgiebigen numerischen Testrechnungen zeigte, kann durch die blockweisen Updates die asymptotische Konvergenzrate deutlich erhöht werden. Zudem kann durch den Einsatz von speziellen, auf blockstrukturierte Probleme zugeschnittenen QP-Lösern weitere Rechenzeit eingespart werden.

2.3.4 Kondensierung

Mit dem folgenden Kondensierungs-Algorithmus kann die Größe des quadratischen Programms deutlich reduziert werden durch ein effizientes an die spezielle Struktur angepaßtes Eliminationsschema. Das resultierende verkleinerte und dicht besetzte QP kann dann mit einem Standard-QP-Löser gelöst werden. Die vollständige Lösung des ursprünglichen QPs erhält man dann durch eine simple Rekursionsformel.

Ausgehend von dem QP $Q(w_k,B_k)$ (2.5) lassen wir zur besseren Übersicht den Iterationsindex k weg und schreiben abkürzend $b := \nabla F(y)$. Weiter fassen wir die linearisierten Stetigkeitsbedingungen zusammen mit $C\Delta w = c$ und die übrigen Beschränkungen mit $D\Delta w\{= \mid \geq\}d$. Wir werden sehen, daß es keinen Unterschied macht, ob die "übrigen" Nebenbedingungen Gleichungen oder Ungleichungen sind. Wir schreiben also das QP $Q(w,B)$ in der Form:

$$\min_{\Delta w'} b^T\Delta w + \tfrac{1}{2}\Delta w B \Delta w \tag{2.14a}$$

$$\left(\frac{C}{D}\right)\Delta w \quad \frac{=}{\{\ =\mid\geq\ \}} \quad \left(\frac{c}{d}\right) \tag{2.14b}$$

Die Matrix (C,D) ist in Abbildung 2.3 dargestellt. Allerdings fehlen in dieser Darstellung noch die Kopplungsbedingungen (1.11g). Diese Bedingungen werden vor der eigentlichen Kondensierung in einer sogenannten *Vorkondensierung* eliminiert.

Wir wollen dies an einem kleinem Beispiel mit nur einer Stufe mit drei Mehrzielintervallen demonstrieren.

Die Jacobi-Matrix (mit Kopplungsbedingungen) unseres Beispielproblems sieht wie folgt aus:

$$\begin{pmatrix}
X^x_{00} & X^z_{00} & X^{p|v}_{00} & X^q_{00} & -I & & & & & & & & & & & \\
 & & I & & & & -I & & & & & & & & & \\
 & & & & X^x_{01} & X^z_{01} & X^{p|v}_{01} & X^q_{01} & -I & & & & & & & \\
 & & I & & & & & & & & -I & & & & & \\
 & & & & & & & & X^x_{02} & X^z_{02} & X^{p|v}_{02} & X^q_{02} & -I & & & \\
 & & I & & & & & & & & & & & & -I & \\
V^x_{00} & V^z_{00} & V^{p|v}_{00} & V^q_{00} & & & & & & & & & & & & \\
 & & & & V^x_{01} & V^z_{01} & V^{p|v}_{01} & V^q_{01} & & & & & & & & \\
 & & & & & & & & V^x_{02} & V^z_{02} & V^{p|v}_{02} & V^q_{02} & & & & \\
 & & & & & & & & & & & & V^x_{03} & V^z_{03} & V^{p|v}_{03} & V^q_{03} \\
R^x_{00} & R^z_{00} & R^{p|v}_{00} & R^q_{00} & R^x_{01} & R^z_{01} & R^{p|v}_{01} & R^q_{01} & R^x_{02} & R^z_{02} & R^{p|v}_{02} & R^q_{02} & R^x_{03} & R^z_{03} & R^{p|v}_{03} & R^q_{03}
\end{pmatrix}$$

Als erstes wählen wir eine Permutation der Variablen, sodaß die Komponenten des Lösungsvektors Δw, die wir eliminieren wollen, am Anfang stehen. Dies sind im Vorkondensierungsschritt $\Delta s^{p|v}_{ij}, (i,j) \neq (0,0)$ und in der eigentlichen Kondensierung $\Delta s^x_{ij}, (i,j) \neq (0,0)$. Der neue Variablenvektor sieht dann wie folgt aus:

$$\Delta w' \;=\; \left(\frac{\Delta w'_1}{\Delta w'_2}\right) \;=\; P\Delta w$$

mit

$$\Delta w_1' := \begin{pmatrix} \Delta(s^x|p|v)_{0,1} \\ \vdots \\ \Delta(s^x|p|v)_{M-1,\hat{m}} \end{pmatrix}$$

$$\Delta w_2' := \begin{pmatrix} \Delta(s^x|s^z|p|v|q)_{0,0} \\ \Delta(s^z|q)_{0,1} \\ \vdots \\ \Delta(s^z|q)_{M-1,\hat{m}} \end{pmatrix}$$

Dabei ist $\hat{m} := m_{M-1}$.
Wir substituieren $\Delta w = P^T \Delta w'$ in das QP $Q(w,B)$ (2.14) und erhalten ein Problem in w':

$$\min_{\Delta w'} (Pb)^T \Delta w' + \tfrac{1}{2} \Delta w' (PBP^T) \Delta w' \tag{2.15a}$$

$$\left(\frac{CP^T}{DP^T}\right) \Delta w' \quad \underset{\{\ =\ |\ \geq\ \}}{=} \quad \left(\frac{c}{d}\right) \tag{2.15b}$$

Für unser Beispiel sieht die Matrix (CP^T, DP^T) wie folgt aus:

$$\begin{pmatrix}
-I & & & & & & X^x_{00} & X^z_{00} & X^{p|v}_{00} & X^q_{00} & & & & & & \\
 & -I & & & & & & & I & & & & & & & \\
X^x_{01} & X^{p|v}_{01} & -I & & & & & & & & X^z_{01} & X^q_{01} & & & & \\
 & & & -I & & & & & I & & & & & & & \\
 & & X^x_{02} & X^{p|v}_{02} & -I & & & & & & & & X^z_{02} & X^q_{02} & & \\
 & & & & & -I & & & I & & & & & & & \\
 & & & & & & V^x_{00} & V^z_{00} & V^{p|v}_{00} & V^q_{00} & & & & & & \\
V^x_{01} & V^{p|v}_{01} & & & & & & & & & V^z_{01} & V^q_{01} & & & & \\
 & & V^x_{02} & V^{p|v}_{02} & & & & & & & & & V^z_{02} & V^q_{02} & & \\
 & & & & V^x_{03} & V^{p|v}_{03} & & & & & & & & & V^z_{03} & V^q_{03} \\
R^x_{01} & R^{p|v}_{01} & R^x_{02} & R^{p|v}_{02} & R^x_{03} & R^{p|v}_{03} & R^x_{00} & R^z_{00} & R^q_{00} & R^{p|v}_{00} & R^z_{01} & R^q_{01} & R^z_{02} & R^q_{02} & R^z_{03} & R^q_{03}
\end{pmatrix}$$

Nun können wir die Variablen $\Delta s^{p|v}_{ij}, (i,j) \neq (0,0)$ eliminieren, und die Matrix kann reduziert werden zu:

$$\begin{pmatrix}
-I & & & X^x_{00} & X^z_{00} & X^{p|v}_{00} & X^q_{00} & & & & & & \\
X^x_{01} & -I & & & & X^{p|v}_{01} & & X^z_{01} & X^q_{01} & & & & \\
 & X^x_{02} & -I & & & X^{p|v}_{02} & & & & X^z_{02} & X^q_{02} & & \\
 & & & V^x_{00} & V^z_{00} & V^{p|v}_{00} & V^q_{00} & & & & & & \\
V^x_{01} & & & & & V^{p|v}_{01} & & V^z_{01} & V^q_{01} & & & & \\
 & V^x_{02} & & & & V^{p|v}_{02} & & & & V^z_{02} & V^q_{02} & & \\
 & & V^x_{03} & & & V^{p|v}_{03} & & & & & & V^z_{03} & V^q_{03} \\
R^x_{01} & R^x_{02} & R^x_{03} & R^x_{00} & R^z_{00} & \sum_{i=0}^{3} R^{p|v}_{0i} & R^q_{00} & R^z_{01} & R^q_{01} & R^z_{02} & R^q_{02} & R^z_{03} & R^q_{03}
\end{pmatrix}$$

Im allgemeinen Fall ist die Struktur der Jacobi-Matrix dann wie in Abbildung 2.4. Hierbei bezeichnet $*$ das "Fill-in", das durch die Vorkondensierung entstanden ist.

$$
\left(\begin{array}{ccccc|ccccccccccccc}
-I & & & & & X^x_{0,0} & X^z_{0,0} & X^{\hat{q}}_{0,0} & & & & & & & & & \\
X^x_{0,1} & -I & & & & & & * & X^z_{0,1} & X^q_{0,1} & & & & & & & \\
 & X^x_{0,2} & & & & & & * & & & X^z_{0,2} & X^q_{0,2} & & & & & \\
 & & \ddots & & & & & \vdots & & & & & \ddots & & & & \\
 & & & X^x_{M-1,\hat{m-1}} & -I & & & * & & & & & & X^z_{M-1,\hat{m-1}} & X^q_{M-1,\hat{m-1}} & & \\
\hline
 & & & & & V^x_{0,0} & V^z_{0,0} & V^{\hat{q}}_{0,0} & & & & & & & & & \\
V^x_{0,1} & & & & & & & * & V^z_{0,1} & V^q_{0,1} & & & & & & & \\
 & V^x_{0,2} & & & & & & * & & & V^z_{0,2} & V^q_{0,2} & & & & & \\
 & & \ddots & & & & & \vdots & & & & & \ddots & & & & \\
 & & & V^x_{M-1,\hat{m-1}} & & & & * & & & & & & V^z_{M-1,\hat{m-1}} & V^q_{M-1,\hat{m-1}} & & \\
 & & & & V^x_{M-1,\hat{m}} & & & * & & & & & & & & V^z_{M-1,\hat{m}} & V^q_{M-1,\hat{m}} \\
\hline
R^x_{0,1} & R^x_{0,2} & \cdots & R^x_{M-1,\hat{m-1}} & R^x_{M-1,\hat{m}} & R^x_{0,0} & R^z_{0,0} & * & R^z_{0,1} & R^q_{0,1} & R^z_{0,2} & R^q_{0,2} & \cdots & R^z_{M-1,\hat{m-1}} & R^q_{M-1,\hat{m-1}} & R^z_{M-1,\hat{m}} & R^q_{M-1,\hat{m}}
\end{array}\right)
$$

Abbildung 2.4: Struktur der Jacobi-Matrix nach der Permutation der Variablen

Die Komponenten des Lösungsvektors Δw, die wir eliminieren wollen, stehen nun am Anfang.

Dann führen wie eine blockweise Gauß-Elimination durch. Analog zur gewöhnlichen Gauß-Elimination zur Dreieckszerlegung von Matrizen läßt sich eine Transformationsmatrix T finden, sodaß:

$$T \left(\begin{array}{c} CP^T \\ \hline DP^T \end{array} \right) = \left(\begin{array}{c|c} -I & C_2' \\ \hline 0 & D_2' \end{array} \right)$$

Wir wenden nun die Transformation T auf die Nebenbedingungen des umsortierten Problems an und erhalten das modifizierte QP $Q'(w,B')$:

$$\min_{\Delta w_1', \Delta w_2'} \left(\begin{array}{c} b_1' \\ \hline b_2' \end{array} \right) \left(\begin{array}{c} \Delta w_1' \\ \hline \Delta w_2' \end{array} \right) + \frac{1}{2} \left(\begin{array}{c} \Delta w_1' \\ \hline \Delta w_2' \end{array} \right)^T \left(\begin{array}{c|c} B_{11}' & B_{12}' \\ \hline B_{12}'^T & B_{22}' \end{array} \right) \left(\begin{array}{c} \Delta w_1' \\ \hline \Delta w_2' \end{array} \right)$$
$$\left(\begin{array}{c|c} -I & C_2' \\ \hline 0 & D_2' \end{array} \right) \left(\begin{array}{c} \Delta w_1' \\ \hline \Delta w_2' \end{array} \right) \quad \begin{array}{c} = \\ \hline \{ = \mid \geq \} \end{array} \quad \left(\begin{array}{c} c' \\ \hline d' \end{array} \right)$$

mit

$$\begin{aligned} b' = \left(\begin{array}{c} b_1' \\ \hline b_2' \end{array} \right) &= Pb \\ B' = \left(\begin{array}{c|c} B_{11}' & B_{12}' \\ \hline B_{12}'^T & B_{22}'^T \end{array} \right) &= PBP^T \\ \left(\begin{array}{c} c' \\ \hline d' \end{array} \right) &= T \left(\begin{array}{c} c \\ \hline d \end{array} \right) \end{aligned}$$

Nun ergibt sich automatisch, daß $\Delta w_1'$ eliminiert werden kann.
Wir substituieren $\Delta w_1' = C_2' \Delta w_2' - c'$ in das Zielfunktional, können die Terme ohne $\Delta w_2'$ vernachlässigen und erhalten so das kondensierte QP $Q''(w,B'')$ von stark reduzierter Größe:

$$\begin{array}{l} \min\limits_{\Delta w_2'} b''^T \Delta w_2' + \frac{1}{2} \Delta w_2'^T B'' \Delta w_2' \\ \left(\begin{array}{c} D_2' \end{array} \right) \Delta w_2' \quad \{ = \mid \geq \} \quad \left(\begin{array}{c} d' \end{array} \right) \end{array}$$

mit der kondensierten Hesse-Matrix

$$B'' = C_2'^T B_{11}' C_2' + C_2'^T B_{12}' + B_{12}'^T C_2' + B_{22}'$$

und dem kondensierten Zielfunktionsgradienten

$$b'' = C_2'^T b_1' + b_2' - C_2'^T B_{11}' c' - B_{12}'^T c'$$

Es ist eine wichtige Beobachtung, daß die Transformationsmatrix T nie explizit ausgerechnet werden muß. Stattdessen können die Matrizen C', D' und die Vektoren c', d' rekursiv berechnet werden.
Das kondensierte quadratische Programm kann durch einen Standard-QP-Löser gelöst werden. Wir verwenden die NAG-Routine E04NAF.
Aus der Lösung des kondensierten QPs kann man sehr effizient durch Rekursionsformeln sowohl die eliminierten Komponenten als auch die Lagrange-Multiplikatoren des ursprünglichen Problems berechnen.

SATZ 2.14 (TRANSFORMATION DER KKT-PUNKTE DURCH DIE KONDENSIERUNG)
Sei $(\Delta w^*,\tilde{\lambda}^*)$ *ein KKT-Punkt des ursprünglichen Problems* $Q(w,B)$*. Dann ist* $(\Delta w'^*, \tilde{\lambda}'^*)$ *mit*

$$\begin{aligned} \Delta w'^* = \begin{pmatrix} \Delta w_1'^* \\ \Delta w_2'^* \end{pmatrix} &= P\Delta w^* \\ \Delta\tilde{\lambda}'^* = \begin{pmatrix} \tilde{\lambda}'^* \\ \tilde{\lambda}'^* \end{pmatrix} &= T^{-T}\tilde{\lambda}^* \end{aligned}$$

ein KKT-Punkt von $Q'(w,B')$ *und* $(\Delta w_2'^*,\tilde{\lambda}_2'^*)$ *ein KKT-Punkt des kondensierten Problems* $Q''(w,B'')$*, wobei die Matrizen* P *und* T *wie oben beschrieben definiert sind.*
Umgekehrt, sei $(\Delta w_2'^*,\tilde{\lambda}_2'^*)$ *ein KKT-Punkt des kondensierten Problems* $Q''(w,B'')$*. Dann ist* $(\Delta w'^*, \tilde{\lambda}'^*)$ *mit*

$$\begin{aligned} \Delta w'^* = \begin{pmatrix} \Delta w_1'^* \\ \Delta w_2'^* \end{pmatrix} &= \begin{pmatrix} C_2'\Delta w_2'^* - c' \\ \Delta w_2'^* \end{pmatrix} \\ \Delta\tilde{\lambda}'^* = \begin{pmatrix} \tilde{\lambda}'^* \\ \tilde{\lambda}'^* \end{pmatrix} &= \begin{pmatrix} -B_{11}'\Delta w_1'^* - B_{12}'\Delta w_2 i^* - b_1' \\ \lambda_2'^* \end{pmatrix} \end{aligned}$$

ein KKT-Punkt von $Q'(w,B')$ *und* $(\Delta w^*, \tilde{\lambda}^*)$ *mit*

$$\begin{aligned} \Delta w^* &= P^T\Delta w'^* \\ \Delta\tilde{\lambda}^* &= T^T\tilde{\lambda}'^* \end{aligned}$$

ein KKT-Punkt des ursprünglichen Problems, wobei die Matrizen C_2'*,* c'*,* B_{11}'*,* B_{12}' *und* b_1' *wie oben beschrieben definiert sind.*

Ein Beweis findet sich in [Lei99]. Er benutzt, daß die Permutation der Variablen durch die Matrix P und die Gauß-Elimination durch die Matrix T lineare Transformationen sind, die die Lagrange-Multiplikatoren $\tilde{\lambda}$ und die Lösung y^* eines QPs invariant lassen.

Eine detaillierte Beschreibung des Kondensierungs-Algorithmus findet man in [Lei95] und [Lei99].

2.3.5 Reduktionstechniken für Probeme der optimalen Steuerung

Die in Abschnitt 2.2.5 vorgestellten reduzierten SQP-Verfahren lassen sich nicht ohne weiteres auf Probleme der optimalen Steuerung anwenden:

- Es ist schwierig und aufwendig, das Verfahren auf Probleme zu verallgemeinern, die nicht nur Gleichungsnebenbedingungen, sondern auch Ungleichungsnebenbedingungen beinhalten.
- Die dem Problem innewohnende Blockstruktur geht verloren und kann bei der Lösung des QPs nicht mehr ausgenutzt werden.

Abhilfe schafft die neue Klasse von reduzierten SQP-Verfahren, die *partiell reduzierten SQP-Verfahren* (PRSQP-Verfahren), die unter diesem Namen erstmals von Schulz ([Sch96]) vorgestellt wurden. Der Reduktionsansatz im SQP-Verfahren wird dabei nur auf einen festen Teil G_2 der Gleichungsnebenbedingungen G angewandt und soll die Stärken von herkömmlichem SQP-Verfahren und den reduzierten Methoden vereinen. Einerseits nutzt man den Vorteil der Reduktion der Gleichungsbeschränkungen, andererseits läßt das Verfahren auch allgemeine Optimierungsprobleme mit Ungleichungsbeschränkungen zu. Vor allem aber können Strukturen und damit Separabilität erhalten werden, indem nur ein Teil der Gleichungsnebenbedingungen für die Zerlegung (2.8) verwenden wird und die Linearisierungen der übrigen Gleichungen und Ungleichungen im QP beibehalten werden.

Wir betrachten das folgende beschränkte Optimierungsproblem, bei dem wir schon explizit eine Unterteilung der Variablen und der Gleichungsnebenbedingungen vorgenommen haben:

$$\min_{(w_1,w_2)} \quad F(w_1,w_2) \tag{2.16}$$

$$\begin{aligned} G_1(w_1,w_2) &= 0 \\ G_2(w_1,w_2) &= 0 \\ H(w_1,w_2) &\geq 0 \end{aligned} \tag{2.17}$$

Hierbei bezeichne $w_1 \in \mathbb{R}^{n_1}$, $w_2 \in \mathbb{R}^{n_2}$, $F \in \mathcal{C}^2(\mathbb{R}^{n_1+n_2},\mathbb{R})$, $G_1 \in \mathcal{C}^2(\mathbb{R}^{n_1+n_2},\mathbb{R}^{m_1})$ mit $m_1 = n_1$, $G_2 \in \mathcal{C}^2(\mathbb{R}^{n_1+n_2},\mathbb{R}^{m_2})$, $H \in \mathcal{C}^2(\mathbb{R}^{n_1+n_2},\mathbb{R}^{m_3})$.
$\nabla_w G_1$ sei nicht-singulär. $S_k^{\mathcal{N}}$, $S_k^{\mathcal{R}}$ sei nun über den Nullraum und den Bildraum von $\nabla_w G_1$ definiert, wie in Abschnitt 2.2.5 beschrieben.
Mit der *Koordinaten-Basis* erhalten wir jetzt

$$\begin{aligned} S_k^{\mathcal{N}} &= \begin{pmatrix} -(\nabla_{w_1} G_{1,k}^T)^{-1} \nabla_{w_2} G_{1,k}^T \\ I \end{pmatrix} \quad \in \mathbb{R}^{n \times (n-m_1)} \\ S_k^{\mathcal{R}} &= \begin{pmatrix} I \\ 0 \end{pmatrix} \quad \in \mathbb{R}^{n \times m_1} \end{aligned}$$

Für diese Wahl der Basen wird die Zerlegung $\Delta w_k = S_k^{\mathcal{R}} y_k^{\mathcal{R}} + S_k^{\mathcal{N}} y_k^{\mathcal{N}}$ zu

$$\begin{pmatrix} \Delta w_{1,k} \\ \Delta w_{2,k} \end{pmatrix} = \begin{pmatrix} -(\nabla_{w_1} G_{1,k}^T)^{-1} \nabla_{w_2} G_{1,k}^T y_k^{\mathcal{N}} + y_k^{\mathcal{R}} \\ y_k^{\mathcal{N}} \end{pmatrix}$$

Der Bildraumschritt ist dann gegeben durch

$$y_k^{\mathcal{R}} = -(\nabla_{w_1} G_{1,k}^T)^{-1} G_{1,k}$$

und der Nullraumschritt ist Lösung des QPs:

$$\begin{aligned} &\min_{y^{\mathcal{N}}} \left(\nabla_{w_2} F_k + D_k^T \nabla_{w1} F_k\right)^T y^{\mathcal{N}} + \tfrac{1}{2} (y^{\mathcal{N}})^T B_k^{\mathcal{N}} y^{\mathcal{N}} \\ G_{2,k} + \nabla_{w1} G_{2,k}^T y_k^{\mathcal{R}} + (\nabla_{w2} G_{2,k}^T + \nabla_{w1} G_{2,k}^T D_k) y^{\mathcal{N}} &= 0 \\ H_k + \nabla_{w1} H_k^T y_k^{\mathcal{R}} + (\nabla_{w2} H_k^T + \nabla_{w1} H_k^T D_k) y^{\mathcal{N}} &\geq 0 \end{aligned}$$

Dabei ist

$$D_k \;:=\; -\left(\nabla_{w1} G_{1,k}^T\right)^{-1} \nabla_{w2} G_{1,k}^T$$

und $B_k^{\mathcal{N}}$ eine Approximation der partiell reduzierten Hesse-Matrix:

$$B_k^{\mathcal{N}} \;\approx\; \left(S_k^{\mathcal{N}}\right)^T \nabla_w^2 \mathcal{L}_k S_k^{\mathcal{N}}$$

Abbildung 2.5-2.7 zeigen das Karush-Kuhn-Tucker-System des reduzierten QPs im Vergleich zu dem KKT-System in einem (vollen) SQP-Verfahren.

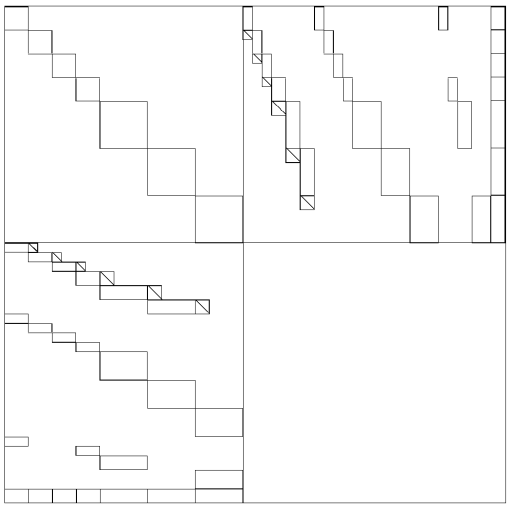

Abbildung 2.5: KKT-Matrix des nicht reduzierten Problems

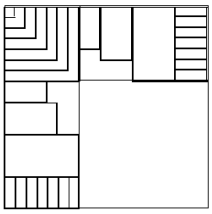

Abbildung 2.6: KKT-Matrix des partiell reduzierten Problems

Das PRSQP-Verfahren mit Update-Formeln ist lokal 2-Schritt-superlinear konvergent.

Wir wenden nun die Technik der PRSQP-Verfahren auf NLPs für große, dünnbesetzte DAE-Modelle vom Index 1, wie wir sie bei den Problemen der optimalen Steuerung haben.

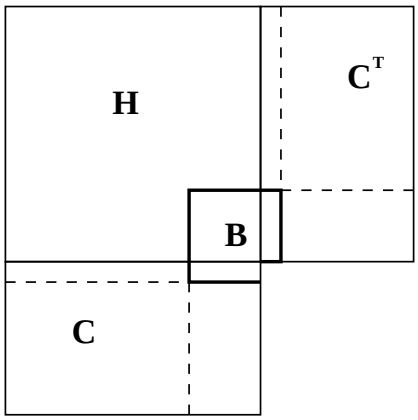

Abbildung 2.7: Vergleich der KKT-Systeme bei SQP bzw. PRSQP

Die Annahme, daß wir nur DAEs vom Index 1 betrachten, führt dazu, daß die algebraischen Variablen komplett bestimmt sind durch die Konsistenzbedingungen für gegebene Werte für die differentiellen Variablen s_{ij}^x und die Parameter $\hat{q}_{ij}$. Dies legt eine natürliche Zerlegung des diskretisierten dynamischen Problems nahe, bei der wir nur die algebraischen Variablen s_{ij}^z als abhängige Variablen im Sinne der partiellen Reduktion sehen.

Basierend auf der Anordnung der Variablen

$$w = \begin{pmatrix} w_{0,0} \\ w_{0,1} \\ \vdots \\ w_{M-1,\hat{m}} \end{pmatrix} \quad \text{mit } w_{ij} := \begin{pmatrix} s_{ij}^x \\ s_{ij}^z \\ \hat{q}_{ij} \end{pmatrix}$$

wählen wir die Matrizen

$$S^{\mathcal{N}} = \begin{pmatrix} S_{0,0}^{\mathcal{N}} & & & \\ & S_{0,1}^{\mathcal{N}} & & \\ & & \ddots & \\ & & & S_{M-1,\hat{m}}^{\mathcal{N}} \end{pmatrix}$$

$$S^{\mathcal{R}} = \begin{pmatrix} S_{0,0}^{\mathcal{R}} & & & \\ & S_{0,1}^{\mathcal{R}} & & \\ & & \ddots & \\ & & & S_{M-1,\hat{m}}^{\mathcal{R}} \end{pmatrix}$$

mit den Diagonalblöcken

$$\begin{aligned} S_{ij}^{\mathcal{N}} &= \begin{pmatrix} I & 0 \\ -(\hat{G}_{ij}^z)^{-1}\hat{G}_{ij}^x & -(\hat{G}_{ij}^z)^{-1}\hat{G}_{ij}^{\hat{q}} \\ 0 & I \end{pmatrix} \\ S_{ij}^{\mathcal{R}} &= \begin{pmatrix} 0 \\ I \\ 0 \end{pmatrix} \end{aligned}$$

Nach Substitution von

$$\Delta w = S^{\mathcal{N}} y^{\mathcal{N}+} + S^{\mathcal{R}} y^{\mathcal{R}}$$

erhalten wir, daß der Bildraumschritt gegeben ist durch

$$y^{\mathcal{R}} = \begin{pmatrix} y_{0,0}^{\mathcal{R}} \\ y_{0,1}^{\mathcal{R}} \\ \vdots \\ y_{M-1,\hat{m}}^{\mathcal{R}} \end{pmatrix} \quad \text{mit } y_{ij}^{\mathcal{R}} = -(\hat{G}_{ij}^z)^{-1}\hat{g}_i(s_{ij}^x, s_{ij}^z, \hat{q}_{ij}, \tau_{ij})$$

und der Nullraumschritt sich darstellen läßt als

$$y^{\mathcal{N}} = \begin{pmatrix} y_{0,0}^{\mathcal{N}} \\ y_{0,1}^{\mathcal{N}} \\ \vdots \\ y_{M-1,\hat{m}}^{\mathcal{N}} \end{pmatrix}$$

wobei

$$y_{ij}^{\mathcal{N}} = \begin{pmatrix} \Delta s_{ij}^x \\ \Delta \hat{q}_{ij} \end{pmatrix}$$

die Lösung des reduzierten QPs ist.
Die Zerlegung, die so entsteht, nutzt einerseits die Dünnbesetztheit der Konsistenzbedingungen und erhält gleichzeitig die Separierbarkeit des originalen Problems. Das resultierende QP hat die gleiche charakteristische Blockstruktur – sowohl in der Hesse-Matrix, als auch in den Jacobi-Matrizen der verbleibenden Nebenbedingungen – wie im Fall der ODEs. Die bereits präsentierten Methoden können direkt übernommen werden.

Wir fassen die Eigenschaften der vorgestellten partiellen Reduktion zusammen:

- Die Zerlegung existiert für alle Index-1-Probleme.
- Wir erhalten eine sinnvolle Reduktion, wenn wir viele algebraische Variablen haben.

- Die Berechnung ist “billig” durch Verwendung von dünnen Faktorisierungen von $\hat{G}_{ij}^z$ und rekursiven Berechnungen der Koordinatenbasis.
- Die Struktur des originalen Problems, die eine Separierbarkeit beinhaltet, bleibt erhalten.
- Die Zerlegung wird gebildet und arbeitet nur auf “lokaler Information” (d.h. die Generierung des QPs ist direkt parallelisierbar).
- Es gibt Möglichkeiten, die Anzahl der für das Aufstellen des reduzierten QPs nötigen Gradientenberechnungen zu reduzieren.

Für Einzelheiten siehe [Lei99] und [Sch04].

2.4 Das Softwarepaket MUSCOD-II

Das Softwarepaket MUSCOD (**MU**ltiple **S**hooting **CO**de for **D**irect optimal control) wurde in seiner ursprünglichen Fassung Anfang der 80er von Plitt unter der Leitung von Bock entwickelt ([Pli81]). Eine wesentlich weiterentwickelte Version (MUSCOD-II) stammt von Leineweber ([LBBS03], [LSBS03]). Weitere Erweiterungen stammen von Schäfer und Diehl. Die Software erlaubt die robuste und effiziente Behandlung von Problemen der optimalen Steuerung basierend auf Index-1-DAEs.

3 Multiple-Setpoint-Probleme

In diesem Abschnitt wird die Problemklasse der Multiple-Setpoint-Probleme vorgestellt. Nach einer kurzen Erläuterung im ersten Abschnitt, was unter dem Begriff “Multiple-Setpoint-Problem” zu verstehen ist, folgen die mathematischen Problemformulierungen, erst für ein allgemeines nichtlineares Multiple-Setpoint-Problem (Abschnitt zwei), dann für Multiple-Setpoint-Probleme der optimalen Steuerung (Abschnitt drei). Nach einer Analyse der besonderen Problemstruktur in Abschnitt vier wird in Abschnitt fünf die Lösung des im SQP-Verfahren auftretenden quadratischen Programms durch ein spezielles Kondensierungs-Verfahren beschrieben, bevor in Abschnitt sechs der SQP-Algorithmus für Multiple-Setpoint-Probleme der optimalen Steuerung formuliert wird.

3.1 Motivation

Wir unterscheiden zwei Arten von Multiple-Setpoint-Optimierungs-Problemen. Einerseits das optimale Design von Systemen mit unterschiedlichen für die Optimierung relevanten Einsatzmöglichkeiten – also simultanes Design für mehrere Prozesse – und andererseits die Optimierung von Systemen für nur einen Prozeß, bei dem aber einzelne System-Parameter unterschiedliche Werte, meist aus einem vorgegebenem Bereich, annehmen können.

Beispiele für den ersten Fall sind das Design von Robotern, die verschiedene Aufgaben erfüllen müssen, die Entwicklung von Reaktionsapparaturen, die für verschiedene Reaktionsabläufe eingesetzt werden sollen, oder der Bau von Maschinen, die verschiedene Werkstoffe verarbeiten müssen. Eine Optimierung muß demnach mehrere komplett verschiedene dynamische Prozesse berücksichtigen. Wir nennen diese verschiedenen Optimierungs-Szenarien *Setpoints*.

Im zweiten Fall haben wir ein oder mehrere Modellparameter, die nicht fix sind. Bei der Modellierung eines Prozesses werden diese System-Parameter meist als fest gewählt. Die bei der Optimierung erhaltene Lösung ist dann die für diesen bestimmten Parametersatz optimale Lösung. In der Praxis läuft der Prozeß jedoch oft unter verschiedenen Rahmenbedingungen ab, was unterschiedliche Gründe haben kann:

- In einer Bereichsoptimierung werden zum Beispiel Turbinen optimiert für einen Betrieb bei unterschiedlichen Anströmgeschwindigkeiten und Winkeln ([Ega99]), oder es werden Motoren optimiert für einen ganzen Drehzahlbereich.
- Bei chemischen Systemen mit nicht vollständig abgeschlossener Parameterschätzung liegen für manche Parameter keine festen Werte vor, sondern es können nur Bereiche angegeben werden, in denen die Werte dieser Parameter liegen.

- In einer statistischen Optimierung sind Parameter im Modell enthalten, die einem statistischen Gesetz folgend schwanken können.

- In der robusten Optimierung sucht man nach einer Lösung, die auch dann noch ihre guten Eigenschaften behält, wenn einzelne System-Parameter gestört werden.

Eine nur für einen Auslegungspunkt optimierte Lösung kann bei Änderung der System-Parameter im vorgegebenem Rahmen nicht nur zu schlechten Ergebnissen im Zielfunktional führen, sondern auch unzulässig werden. Es wäre also wünschwenswert, eine Lösung zu erhalten, die das Problem nicht nur für einen gegebenen Parametersatz optimiert, sondern für ganze Bereiche der Parameter unter Einhaltung der Nebenbedingungen “optimal” löst. Man wählt dazu verschiedene Referenzszenarien aus und startet eine simultane Optimierung der einzelnen Setpoints.

Für eine simultane Optimierung von verschiedenen Setpoints muß einerseits eine Reihe von Modellgleichungen, die unabhängig voneinander sind, gelöst werden und andererseits eine Zielfunktion formuliert werden, die ein Kriterium beschreibt für eine “globale” Verbesserung in allen Setpoints.

Abbildung 3.1 zeigt den Unterschied zwischen einem Turbinenschaufel-Profil, das für einen Betriebspunkt optimiert wurde, und einem für einen Betriebsbereich optimierten Profil bei variablen Anströmwinkeln ([Ega99]).

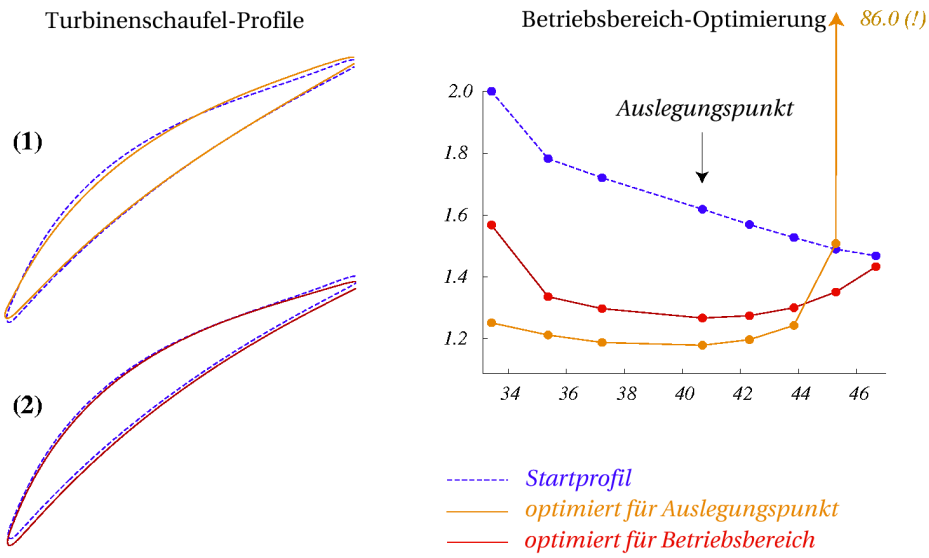

Abbildung 3.1: Betriebsbereich-Optimierung einer Turbinenschaufel

3.2 Allgemeines nichtlineares Multiple-Setpoint-Problem

Wir leiten zunächst die Formulierung für ein allgemeines nichtlineares Multiple-Setpoint-Problem her. Das Single-Setpoint-Problem, von dem wir ausgehen, ist das in Abschnitt 2.1.1 vorgestellte Problem (2.1):

$$\begin{array}{rcl} \min\limits_{w \in D} & F(w) & \\ G(w) & = & 0 \\ H(w) & \geq & 0 \end{array}$$

Oft lassen sich die Variablen w unterteilen in freie Optimierungsparameter p und durch die Gleichungen G festgelegte Variablen x:

$$\begin{array}{rcl} \min\limits_{x,p} & F(x,p) & \\ G(x,p) & = & 0 \\ H(x,p) & \geq & 0 \end{array}$$

mit $\frac{\partial G}{\partial x}$ invertierbar.

Das entsprechende Multiple-Setpoint-Problem setzt sich aus einer Anzahl von N Teilproblemen dieser Art zusammen, die durch eine Zielfunktion gekoppelt sind, die ein Maß für die Verbesserung der Zielfunktionen der einzelnen Setpoints angibt. Wir unterscheiden nun zwischen setpointspezifischen Optimierungs-Parametern p_i und "globalen" Optimierungs-Parametern $\tilde{p}$.

$$\min_{x_i,p_i,\tilde{p}} \sum_{i=0}^{N-1} \omega_i F_i(x_i,p_i,\tilde{p})$$

$$\left.\begin{array}{rcl} G_i(x_i,p_i,\tilde{p}) & = & 0 \\ H_i(x_i,p_i,\tilde{p}) & \geq & 0 \end{array}\right\} \text{für } i = 0,\ldots,N-1$$

mit $\frac{\partial G_i}{\partial x_i}$ invertierbar.
Für manche Probleme ist es notwendig, zusätzliche Nebenbedingungen zu formulieren, die die Setpoints koppeln:

$$\begin{array}{rcl} \sum\limits_{i=0}^{N-1} \tilde{G}_i(x_i,p_i,\tilde{p}) & = & 0 \\ \sum\limits_{i=0}^{N-1} \tilde{H}_i(x_i,p_i,\tilde{p}) & \geq & 0 \end{array}$$

Wir wählen als Zielfunktion des Multiple-Setpoint-Problems eine gewichtete Summe der Zielfunktionen der einzelnen Setpoints, was den Vorteil mit sich bringt, daß auch in den Ableitungen die Separabilität erhalten bleibt (siehe Abschnitt 3.4).

Denkbar wäre auch eine gewichtete Quadratsumme:

$$\min_{x_i,p_i,\tilde{p}} \sum_{i=0}^{N-1} \omega_i F_i(x_i,p_i,\tilde{p})^2$$

Dies läßt sich aber auch einfach dadurch realisieren, daß die Zielfunktionen F_i der einzelnen Setpoints ersetzt werden durch $\tilde{F}_i(x_i,p_i,\tilde{p}) := F_i(x_i,p_i,\tilde{p})^2$, sodaß wir bei der Formulierung als gewichtete Summe bleiben wollen.
Eine Multiple-Setpoint-Zielfunktion in Form einer min-max-Optimierung

$$\min_{x_i,p_i,\tilde{p}} \max_i F_i(x_i,p_i,\tilde{p}) \tag{3.1}$$

läßt sich ebenfalls realisieren mithilfe der folgenden Darstellung:

$$\min_{x_i,p_i,\tilde{p}} \hat{F}$$

mit zusätzlichen Ungleichungen:

$$F_i(x_i,p_i,\tilde{p}) - \hat{F} \leq 0 \quad \forall i = 0,\ldots,N-1$$

Durch diese Formulierung wird außerdem das Problem der Unstetigkeit der Zielfunktion (3.1) umgangen.

3.3 Multiple-Setpoint-Problem der optimalen Steuerung

Wir wollen nun eine Formulierung eines Multiple-Setpoint-Problems der optimalen Steuerung (1.5) herleiten. Zur besseren Übersicht betrachten wir nur Modelle mit einer Stufe und einem Mayer-Zielfunktional.
Wir gehen davon aus, daß jeder Setpoint $i = 0,\ldots,N-1$ seine eigenen dynamischen Variablen x_i und seine eigene Dynamik f_i,g_i hat. Es seien $[0,t_i]$ die Zeithorizonte, in denen die Dynamik der einzelnen Setpoints abläuft.
Oft unterscheiden sich die einzelnen Modelle nur um einen Parameter, der verschiedene Werte α_i annimmt:

$$\begin{aligned} f_i(\cdot) &= f(x(t),u(t),p,t,\alpha_i) \\ g_i(\cdot) &= g(x(t),u(t),p,t,\alpha_i) \end{aligned}$$

Bei den Pfadbeschränkungen können entweder die gleichen Bedingungen $h(x_i)$ für alle x_i gelten oder auch unterschiedliche Bedingungen $h_i(x_i)$ für jeden Setpoint. Wir wählen die allgemeinere Formulierung mit $h_i(x_i)$.
Bei den Randbedingungen wollen wir sowohl setpointspezifische Bedingungen r als auch globale Bedingungen $\tilde{r}$ formulieren, um auch Probleme betrachten zu können, bei denen es Kopplungen zwischen den einzelnen Setpoints gibt.

Bei den Parametern unterscheiden wir zwischen setpointspezifischen (p_i) und globalen ($\tilde{p}$) Größen.
Was die Steuerungen angeht, unterscheiden wir zwischen zwei Typen von Problemen. Erstens Probleme, bei denen die Dynamik bei allen Setpoints im gleichen Zeithorizont abläuft ($\tilde{t} = t_i$) und bei denen es neben eventuellen setpointspezifischen Steuerungen u_{ij} auch globale Steuerungen $\tilde{u}_j$ gibt. Und zweitens Probleme, bei denen die Zeithorizonte unabhängig voneinander sind. Bei diesen Problemen sind globale Steuerungen nicht sinnvoll. (Siehe Abbildung 3.2)
Weiterhin wollen wir zusätzliche Bedingungen einführen, die nötig sind, falls gleiche Anfangswerte bei allen Setpoints gefordert sind.

Es ergeben sich folgende Problemformulierungen:

[A] Multiple-Setpoint-Problem der optimalen Steuerung mit gleichen Zeithorizonten bei allen Setpoints:

$$\min_{x_i(t),z_i(t),u_i(t),p_i,\tilde{u}(t),\tilde{p},\tilde{t}} \sum_{i=0}^{N-1} \omega_i \Phi_i(x_i(\tilde{t}),z_i(\tilde{t}),p_i,\tilde{p},\tilde{t})$$

unter den folgenden Nebenbedingungen:

DAE-Modellgleichungen

$$\left.\begin{array}{rcl} A_i(\cdot)\dot{x}_i(t) & = & f_i(x_i(t),z_i(t),u_i(t),\tilde{u}(t),p_i,\tilde{p},t) \\ 0 & = & g_i(x_i(t),z_i(t),u_i(t),\tilde{u}(t),p_i,\tilde{p},t) \end{array}\right\} t \in [0,\tilde{t}]$$

Pfadbeschränkungen

$$h_i(x_i(t),u_i(t),\tilde{u}(t),p_i,\tilde{p},t) \;\geq\; 0, \quad t \in [0,\tilde{t}]$$

Randbedingungen

$$r_i^s(x_i(0),z_i(0),p_i,\tilde{p},0) + r_i^e(x_i(\tilde{t}),z_i(\tilde{t}),p_i,\tilde{p},\tilde{t}) \quad \left\{ \begin{array}{c} = \\ \geq \end{array} \right\} \quad 0$$

Gekoppelte Randbedingungen

$$\sum_{i=0}^{N-1} \tilde{r}_i^s(x_i(0),z_i(0),p_i,\tilde{p},0) + \tilde{r}_i^e(x_i(\tilde{t}),z_i(\tilde{t}),p_i,\tilde{p},\tilde{t}) \quad \left\{ \begin{array}{c} = \\ \geq \end{array} \right\} \quad 0$$

Kopplung der Anfangswerte

$$x_i(0) \;=\; x_0(0) \qquad \text{für } i = 1,\ldots,N-1$$

} für $i = 0,\ldots,N-1$

[B] Multiple-Setpoint-Problem der optimalen Steuerung mit unterschiedlichen Zeithorizonten bei den einzelnen Setpoints:

$$\min_{x_i(t),z_i(t),u_i(t),p_i,\tilde{p},t_i} \sum_{i=0}^{N-1} \omega_i \Phi_i(x_i(t_i),z_i(t_i),p_i,\tilde{p},t_i)$$

unter den folgenden Nebenbedingungen:

DAE-Modellgleichungen

$$\left.\begin{array}{rcl} A_i(\cdot)\dot{x}_i(t) & = & f_i(x_i(t),z_i(t),u_i(t),p_i,\tilde{p},t) \\ 0 & = & g_i(x_i(t),z_i(t),u_i(t),p_i,\tilde{p},t) \end{array}\right\} t \in [0,t_i]$$

Pfadbeschränkungen

$$h_i(x_i(t),u_i(t),p_i,\tilde{p},t) \geq 0, \quad t \in [0,t_i]$$

Randbedingungen

$$r_i^s(x_i(0),z_i(0),p_i,\tilde{p},0) + r_i^e(x_i(t_i),z_i(t_i),p_i,\tilde{p},t_i) \left\{ \begin{array}{c} = \\ \geq \end{array} \right\} 0$$

Gekoppelte Randbedingungen

$$\sum_{i=0}^{N-1} \tilde{r}_i^s(x_i(0),z_i(0),p_i,\tilde{p},0) + \tilde{r}_i^e(x_i(t_i),z_i(t_i),p_i,\tilde{p},t_i) \left\{ \begin{array}{c} = \\ \geq \end{array} \right\} 0$$

Kopplung der Anfangswerte

$$x_i(0) = x_0(0) \quad \text{für } i = 1,\ldots,N-1$$

} für $i = 0,\ldots,N-1$

Das unendlich-dimensionale Multiple-Setpoint-Problem läßt sich durch die in Abschnitt 1.3 beschriebene Diskretisierung auf eine zu (1.11) entsprechende Form bringen.
Wie bei der Formulierung des Single-Setpoint-Problems verwenden wir eine "Lokalisierung" aller Variablen auf Mehrzielintervallebene, um anschließend ein entkoppeltes Problem und eine separierbare Lagrange-Funktion zu erhalten. Wir vereinen jetzt die beiden Problemtypen durch folgende Schreibweisen:

- Wir verwenden nur noch die lokalisierte Variable v_{ij} und geeignete Kopplungsbedingungen für identische bzw. verschiedene Zeithorizonte auf den einzelnen Setpoints.
- Wir verwenden setpointspezifische und globale diskretisierte Steuerungen q_{ij} bzw. $\tilde{q}_j$ im Wissen, daß setpointspezifische Steuerungen nur Sinn machen, wenn die Zeithorizonte und das Mehrzielgitter auf den verschiedenen Setpoints identisch sind.

Die diskretisierten Problemformulierungen für die beiden Typen von Multiple-Setpoint-Problemen der optimalen Steuerung unterscheiden sich dann nur noch in den Kopplungsbedingungen:
Kopplung der setpointspezifischen Parameter

$$p_{i,j} \;=\; p_{i,0} \quad \forall (i,j), j > 0$$

Kopplung der globalen Parameter

$$\tilde{p}_{i,j} \;=\; \tilde{p}_{0,0} \quad \forall (i,j) \neq (0,0)$$

Kopplung der Zeithorizonte für alle Setpoints (Typ [A]) bzw. für einzelne Setpoints (Typ [B])

$$v_{i,j} \;=\; v_{0,0} \quad \forall (i,j) \neq (0,0)$$

bzw.

$$v_{i,j} \;=\; v_{i,0} \quad \forall (i,j), j > 0$$

Kopplung der Steuerungen für alle Setpoints (Typ [B])

$$\tilde{q}_{i,j} \;=\; \tilde{q}_{0,j} \quad \forall (i,j), i > 0$$

Kopplung der Startwerte für alle Setpoints

$$s^x_{i,0} \;=\; s^x_{0,0} \quad \forall i > 0$$

Abbildung 3.2 zeigt die Struktur der setpointspezifischen bzw. globalen Parametern und (diskretisierten) Steuerungen.

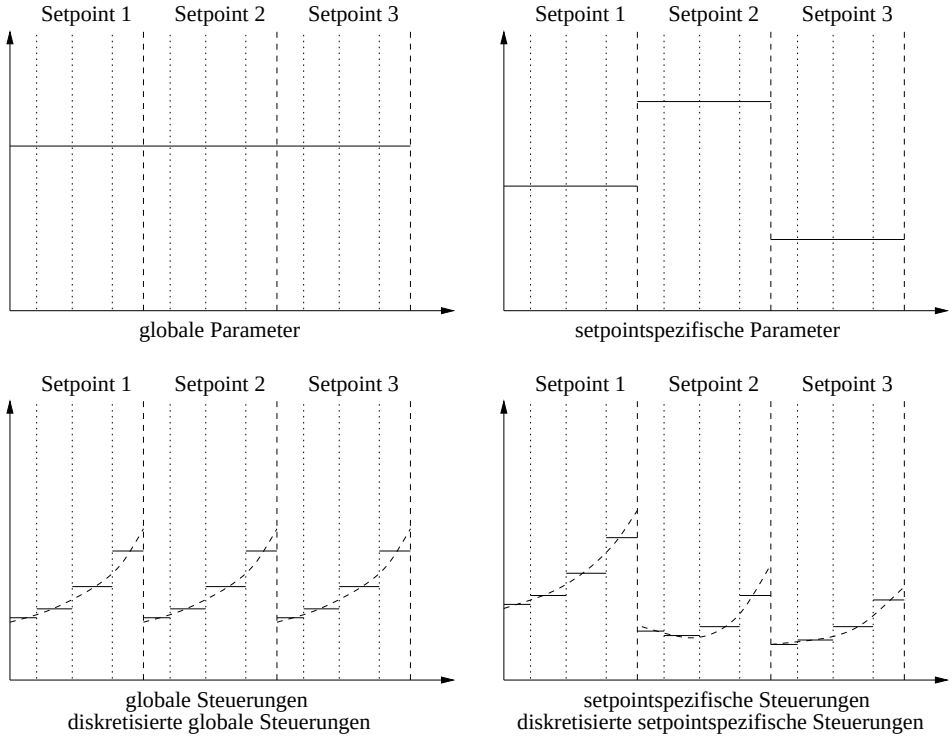

Abbildung 3.2: Setpointspezifische bzw. globale Parameter und Steuerungen

Aus der Diskretisierung des Multiple-Setpoint-Problems der optimalen Steuerung ergibt sich folgendes Multiple-Setpoint-NLP:

$$\min_{s^x_{ij},s^z_{ij},q_{ij},p_i,\tilde{q}_j,\tilde{p}} \sum_{i=0}^{N-1} \omega_i \Phi_i(s^x_{i,m_i},s^z_{i,m_i},p_i,\tilde{p},\theta_i(\tau_{i,m_i},v))$$

unter den folgenden Nebenbedingungen:

Stetigkeitsbedingungen

$$x_i(t_{i,j+1}; s^x_{ij},s^z_{ij},q_{ij},p_i,\tilde{q}_j,\tilde{p},v_i) - s^x_{i,j+1} = 0 \quad j = 0,\ldots,m_i-1$$

Konsistenzbedingungen

$$g_i(s^x_{ij},s^z_{ij},\chi_{ij}(\tau_{ij},q_{ij},\tilde{q}_j),p,\tilde{p}_i,\theta_i(\tau_{ij},v_i)) = 0 \quad j = 0,\ldots,m_i$$

Pfadbeschränkungen

$$h_i(s^x_{ij},s^z_{ij},\chi_{ij}(\tau_{ij},q_{ij},\tilde{q}_j),p,\tilde{p}_i,\theta_i(\tau_{ij},v_i)) \geq 0 \quad j = 0,\ldots,m_i$$

Mehrpunktbedingungen

$$\sum_{j=0}^{m_i-1} r_{ij}(s^x_{ij},s^z_{ij},p,\tilde{p}_i,\theta_i(\tau_{ij},v_i)) \left\{ \begin{array}{c} = \\ \geq \end{array} \right\} 0$$

} für $i = 0,\ldots,N-1$

Gekoppelte Mehrpunktbedingungen

$$\sum_{i=0}^{N-1}\sum_{j=0}^{m_i-1} \tilde{r}_{ij}(s^x_{ij},s^z_{ij},p,\tilde{p}_i,\theta_i(\tau_{ij},v_i)) \left\{ \begin{array}{c} = \\ \geq \end{array} \right\} 0$$

Kopplungsbedingungen (Typ [A] oder [B])

$$\begin{array}{lcll} p_{i,j} = p_{i,0} & \forall (i,j), j>0 & \text{bzw.} & p_{i,j} = p_{i,0} \quad \forall (i,j), j>0 \\ \tilde{p}_{i,j} = \tilde{p}_{0,0} & \forall (i,j) \neq (0,0) & & \tilde{p}_{i,j} = \tilde{p}_{0,0} \quad \forall (i,j) \neq (0,0) \\ v_{i,j} = v_{0,0} & \forall (i,j) \neq (0,0) & & v_{i,j} = v_{i,0} \quad \forall (i,j), j>0 \\ \tilde{q}_{i,j} = \tilde{q}_{0,j} & \forall (i,j), i>0 & & \\ s^x_{i,0} = s^x_{0,0} & \forall i>0 & & s^x_{i,0} = s^x_{0,0} \quad \forall i>0 \end{array}$$

3.4 Problemstruktur

Im folgenden beschreiben wir die Struktur der einzelnen Elemente der Multiple-Setpoint-Problemformulierung aus dem vorhergehenden Abschnitt.

Wir verwenden folgende Anordnung der Variablen:

$$w = \begin{pmatrix} w_{0,0} \\ w_{0,1} \\ \vdots \\ w_{N-1,\hat{m}} \end{pmatrix} \quad \text{mit } w_{ij} := \begin{pmatrix} s_{ij}^x \\ s_{ij}^z \\ \hat{q}_{ij} \end{pmatrix}, \hat{q}_{ij} := \begin{pmatrix} p_{ij} \\ q_{ij} \\ \tilde{p}_{ij} \\ \tilde{q}_{ij} \\ v_{ij} \end{pmatrix}, \hat{m} := m_{N-1}$$

Man beachte, daß am Endpunkt des letzten Mehrzielintervalls in jedem Setpoint keine Steuerungen definiert sind.
Als Zielfunktion hatten wir eine gewichtete Summe gewählt (siehe Abschnitt 3.2), um auch in den Ableitungen die Separabilität zu erhalten:

$$\begin{aligned} F(\cdot) &= \sum_{i=0}^{N-1} \omega_i F_i(\cdot) \\ \nabla_w F(\cdot) &= \sum_{i=0}^{N-1} \omega_i \nabla_w F_i(\cdot) \\ \nabla_w^2 F(\cdot) &= \sum_{i=0}^{N-1} \omega_i \nabla_w^2 F_i(\cdot) \end{aligned}$$

Da auch die Nebenbedingungen Blockstruktur in Hinblick auf die einzelnen Setpoints haben, überträgt sich die Separabilität auf die Lagrange-Funktion:

$$\begin{aligned} \mathcal{L}(w,\lambda,\mu) &= F(w) - \lambda^T G(w) - \mu^T H(w) = \sum_{i=0}^{N-1} \mathcal{L}_i(w,\lambda,\mu) \\ \nabla\mathcal{L}(w,\lambda,\mu) &= \nabla F(w) - \lambda^T \nabla G(w) - \mu^T \nabla H(w) = \begin{pmatrix} \nabla_{w_0}\mathcal{L}_0(w,\lambda,\mu) \\ \vdots \\ \nabla_{w_{N-1}}\mathcal{L}_{N-1}(w,\lambda,\mu) \end{pmatrix} \\ \nabla^2\mathcal{L}(w,\lambda,\mu) &= \begin{pmatrix} \mathcal{B}_{0,0} & & & \\ & \mathcal{B}_{0,1} & & \\ & & \ddots & \\ & & & \mathcal{B}_{N-1,\hat{m}} \end{pmatrix} \quad \text{mit } \mathcal{B}_{ij} := \nabla_{w_{ij}}^2 \mathcal{L}_{ij}(w_{ij},\lambda,\mu) \end{aligned}$$

Die Jacobi-Matrix der Stetigkeitsbedingungen lautet:

$$X = \begin{pmatrix} \tilde{X}_0 & & \\ & \ddots & \\ & & \tilde{X}_{N-1} \end{pmatrix}$$

mit

$$\tilde{X}_i = \begin{pmatrix} X^x_{i,0} & X^z_{i,0} & X^{\hat{q}}_{i,0} & -I & & & & & & & & \\ & & & X^x_{i,1} & X^z_{i,1} & X^{\hat{q}}_{i,1} & -I & & & & & \\ & & & & & & \ddots & & & & & \\ & & & & & & & X^x_{i,m_i-1} & X^z_{i,m_i-1} & X^{\hat{q}}_{i,m_i-1} & -I & 0 & 0 \end{pmatrix}$$

für $i = 0, \ldots, N-1$.
Die Jacobi-Matrix der Konsistenzbedingungen und der diskretisierten Steuerungs- und Zustandsbeschränkungen lautet:

$$V = \begin{pmatrix} V^x_{0,0} & V^z_{0,0} & V^{\hat{q}}_{0,0} & & & & & & & \\ & & & V^x_{0,1} & V^z_{0,1} & V^{\hat{q}}_{0,1} & & & & \\ & & & & & & \ddots & & & \\ & & & & & & & V^x_{N-1,\hat{m}} & V^z_{N-1,\hat{m}} & V^{\hat{q}}_{N-1,\hat{m}} \end{pmatrix}$$

Die Jacobi-Matrix der setpointspezifischen Mehrpunktbedingungen lautet:

$$\begin{pmatrix} \tilde{R}_0 & & \\ & \ddots & \\ & & \tilde{R}_N \end{pmatrix}$$

mit

$$\tilde{R}_i = \begin{pmatrix} R^x_{i,0} & R^z_{i,0} & R^{\hat{q}}_{i,0} & \ldots & R^x_{i,m_i-1} & R^z_{i,m_i-1} & R^{\hat{q}}_{i,m_i-1} \end{pmatrix}$$

für $i = 0, \ldots, N-1$.

Die Jacobi-Matrix der "globalen" Mehrpunktbedingungen lautet:

$$\begin{pmatrix} \tilde{R}^x_{0,0} & \tilde{R}^z_{0,0} & \tilde{R}^{\hat{q}}_{0,0} & \ldots & \ldots \tilde{R}^x_{N-1,\hat{m}} & \tilde{R}^z_{N-1,\hat{m}} & \tilde{R}^{\hat{q}}_{N-1,\hat{m}} \end{pmatrix}$$

3.5 Lösung des Multiple-Setpoint-QPs

Dieser Abschnitt beschreibt ein auf das Multiple-Setpint-Problem zugeschnittenes Kondensierungs-Verfahren zur effizienten Lösung des strukturierten QPs. Ziel ist es wieder, Variablen zu eliminieren. Wir wählen also eine Permutation der Variablen, sodaß die Komponenten des Lösungsvektors Δw, die wir eliminieren wollen, am Anfang stehen.
Im Vorkondensierungsschritt sind dies die folgenden Variablen:

$$\begin{array}{ll} \Delta p_{ij} & \forall (i,j), j > 0 \\ \Delta \tilde{p}_{ij} & \forall (i,j) \neq (0,0) \\ \Delta v_{ij} & \forall (i,j) \neq (0,0) \\ \Delta \tilde{q}_{ij} & \forall (i,j), i > 0 \\ \Delta s^x_{i,0} & \forall i > 0 \end{array}$$

für Problemtyp [A] und

$$\begin{array}{ll} \Delta p_{ij} & \forall (i,j), j > 0 \\ \Delta \tilde{p}_{ij} & \forall (i,j) \neq (0,0) \\ \Delta v_{ij} & \forall (i,j), j > 0 \\ \Delta s^x_{i,0} & \forall i > 0 \end{array}$$

für Problemtyp [B].

In der eigentlichen Kondensierung sind es die Variablen

$$\Delta s^x_{ij} \quad \forall (i,j), j > 0$$

Wir wollen die Vorkondensierungsschritte an einem Beispiel mit nur zwei Setpoints mit jeweils zwei Mehrzielintervallen demonstrieren. Wir betrachten die Vorkondensierung für die verschiedenen Kopplungen separat und lassen zur besseren Übersicht einige Spalten der Jacobi-Matrix weg. Wir betrachten jeweils nur die Spalten, die zu den Variablen Δs^x_{ij} und zu den entsprechenden Kopplungbedingungen gehören.

Wir beginnen mit den Kopplungsbedingungen für $\tilde{p}$ und $\tilde{v}$. Die (vereinfachte) Jacobi-Matrix unseres Beispielproblems sieht wie folgt aus:

$$\left(\begin{array}{cccccccccccc}
X^x_{00} & X^p_{00} & -I & & & & & & & & & \\
 & I & & -I & & & & & & & & \\
 & & X^x_{01} & X^p_{01} & -I & & & & & & & \\
 & I & & & & -I & & & & & & \\
 & & & & & & X^x_{10} & X^p_{10} & -I & & & \\
 & I & & & & & & & & -I & & \\
 & & & & & & & & X^x_{11} & X^p_{11} & -I & \\
 & I & & & & & & & & & & -I \\
\hline
V^x_{00} & V^p_{00} & & & & & & & & & & \\
 & & V^x_{01} & V^p_{01} & & & & & & & & \\
 & & & & V^x_{02} & V^p_{02} & & & & & & \\
 & & & & & & V^x_{10} & V^p_{10} & & & & \\
 & & & & & & & & V^x_{11} & V^p_{11} & & \\
 & & & & & & & & & & V^x_{12} & V^p_{12} \\
\hline
R^x_{00} & R^p_{00} & R^x_{01} & R^p_{01} & R^x_{02} & R^p_{02} & & & & & & \\
 & & & & & & R^x_{10} & R^p_{10} & R^x_{11} & R^p_{11} & R^x_{12} & R^p_{12} \\
\tilde{R}^x_{00} & \tilde{R}^p_{00} & \tilde{R}^x_{01} & \tilde{R}^p_{01} & \tilde{R}^x_{02} & \tilde{R}^p_{02} & \tilde{R}^x_{10} & \tilde{R}^p_{10} & \tilde{R}^x_{11} & \tilde{R}^p_{11} & \tilde{R}^x_{12} & \tilde{R}^p_{12}
\end{array}\right)$$

Nach Umsortierung und Elimination der Variablen $\Delta s_{ij}^{\tilde{p}}, \forall (i,j) \neq (0,0)$ analog zur Prozedur in Abschnitt 2.3.4 ergibt sich:

$$\left(\begin{array}{cccccccc}
-I & & & & X_{00}^x & X_{00}^p & & \\
X_{01}^x & -I & & & & X_{01}^p & & \\
 & & -I & & & X_{10}^p & X_{10}^x & \\
 & & X_{11}^x & -I & & X_{11}^p & & \\
\hline
 & & & & V_{00}^x & V_{00}^p & & \\
V_{01}^x & & & & & V_{01}^p & & \\
 & V_{02}^x & & & & V_{02}^p & & \\
 & & & & & V_{10}^p & V_{10}^x & \\
 & & V_{11}^x & & & V_{11}^p & & \\
 & & & V_{12}^x & & V_{12}^p & & \\
\hline
R_{01}^x & R_{02}^x & & & R_{00}^x & R_{00}^p{+}R_{01}^p{+}R_{02}^p & & \\
 & & R_{11}^x & R_{12}^x & & R_{10}^p{+}R_{11}^p{+}R_{12}^p & R_{10}^x & \\
\tilde{R}_{01}^x & \tilde{R}_{02}^x & \tilde{R}_{11}^x & \tilde{R}_{12}^x & \tilde{R}_{00}^x & \tilde{R}_{00}^p{+}\ldots{+}\tilde{R}_{12}^p & \tilde{R}_{10}^x &
\end{array}\right)$$

Als nächstes untersuchen wir die Kopplungsbedingungen für p und v. Die (vereinfachte) Jacobi-Matrix unseres Beispielproblems sieht wie folgt aus:

$$\left(\begin{array}{cccccccccccc}
X_{00}^x & X_{00}^p & -I & & & & & & & & & \\
 & I & & -I & & & & & & & & \\
 & & X_{01}^x & X_{01}^p & -I & & & & & & & \\
 & I & & & & -I & & & & & & \\
 & & & & & & X_{10}^x & X_{10}^p & -I & & & \\
 & & & & & & & I & & -I & & \\
 & & & & & & & & X_{11}^x & X_{11}^p & -I & \\
 & & & & & & & I & & & & -I \\
\hline
V_{00}^x & V_{00}^p & & & & & & & & & & \\
 & & V_{01}^x & V_{01}^p & & & & & & & & \\
 & & & & V_{02}^x & V_{02}^p & & & & & & \\
 & & & & & & V_{10}^x & V_{10}^p & & & & \\
 & & & & & & & & V_{11}^x & V_{11}^p & & \\
 & & & & & & & & & & V_{12}^x & V_{12}^p \\
\hline
R_{00}^x & R_{00}^p & R_{01}^x & R_{01}^p & R_{02}^x & R_{02}^p & & & & & & \\
 & & & & & & R_{10}^x & R_{10}^p & R_{11}^x & R_{11}^p & R_{12}^x & R_{12}^p \\
\tilde{R}_{00}^x & \tilde{R}_{00}^p & \tilde{R}_{01}^x & \tilde{R}_{01}^p & \tilde{R}_{02}^x & \tilde{R}_{02}^p & \tilde{R}_{10}^x & \tilde{R}_{10}^p & \tilde{R}_{11}^x & \tilde{R}_{11}^p & \tilde{R}_{12}^x & \tilde{R}_{12}^p
\end{array}\right)$$

Nach Umsortierung und Elimination der Variablen $\Delta s_{ij}^p, \forall(i,j), j > 0$ ergibt sich:

$$\left(\begin{array}{cccccccc}
-I & & & & X_{00}^x & X_{00}^p & & \\
X_{01}^x & -I & & & & X_{01}^p & & \\
 & & -I & & & & X_{10}^x & X_{10}^p \\
 & & X_{11}^x & -I & & & & X_{11}^p \\
\hline
 & & & & V_{00}^x & V_{00}^p & & \\
V_{01}^x & & & & & V_{01}^p & & \\
 & V_{02}^x & & & & V_{02}^p & & \\
 & & & & & & V_{10}^x & V_{10}^p \\
 & & V_{11}^x & & & & & V_{11}^p \\
 & & & V_{12}^x & & & & V_{12}^p \\
\hline
R_{01}^x & R_{02}^x & & & R_{00}^x & R_{00}^p+R_{01}^p+R_{02}^p & & \\
 & & & R_{12}^x & & R_{11}^x & R_{10}^x & R_{10}^p+R_{11}^p+R_{12}^p \\
\tilde{R}_{01}^x & \tilde{R}_{02}^x & \tilde{R}_{11}^x & \tilde{R}_{12}^x & \tilde{R}_{00}^x & \tilde{R}_{00}^p+\tilde{R}_{01}^p+\tilde{R}_{02}^p & \tilde{R}_{10}^x & \tilde{R}_{10}^p+\tilde{R}_{11}^p+\tilde{R}_{12}^p
\end{array}\right)$$

Als letztes untersuchen die Kopplungsbedingungen für $\tilde{q}$. Die (vereinfachte) Jacobi-Matrix unseres Beispielproblems sieht wie folgt aus:

$$\left(\begin{array}{cccccccccccc}
X_{00}^x & X_{00}^q & -I & & & & & & & & & \\
 & & X_{01}^x & X_{01}^q & -I & & & & & & & \\
 & I & & & & & & -I & & & & \\
 & & & & & & X_{10}^x & X_{10}^q & -I & & & \\
 & & & I & & & & & & -I & & \\
 & & & & & & & & X_{11}^x & X_{11}^q & -I & \\
 & & & & & I & & & & & & -I \\
\hline
V_{00}^x & V_{00}^q & & & & & & & & & & \\
 & & V_{01}^x & V_{01}^q & & & & & & & & \\
 & & & & V_{02}^x & V_{02}^q & & & & & & \\
 & & & & & & V_{10}^x & V_{10}^q & & & & \\
 & & & & & & & & V_{11}^x & V_{11}^q & & \\
 & & & & & & & & & & V_{12}^x & V_{12}^q \\
\hline
R_{00}^x & R_{00}^p & R_{01}^x & R_{01}^p & R_{02}^x & R_{02}^p & & & & & & \\
 & & & & & & R_{10}^x & R_{10}^p & R_{11}^x & R_{11}^p & R_{12}^x & R_{12}^p \\
\tilde{R}_{00}^x & \tilde{R}_{00}^q & \tilde{R}_{01}^x & \tilde{R}_{01}^q & \tilde{R}_{02}^x & \tilde{R}_{02}^q & \tilde{R}_{10}^x & \tilde{R}_{10}^q & \tilde{R}_{11}^x & \tilde{R}_{11}^q & \tilde{R}_{12}^x & \tilde{R}_{12}^q
\end{array}\right)$$

Nach Umsortierung und Elimination der Variablen $\Delta\tilde{q}_{ij}, \forall(i,j), i > 0$ ergibt sich:

$$\left(\begin{array}{ccccccccc}
-I & & & & X^x_{00} & X^q_{00} & & & \\
X^x_{01} & -I & & & & & X^q_{01} & & \\
 & & -I & & & X^q_{10} & & & X^x_{10} \\
 & & X^x_{11} & -I & & & X^q_{11} & & \\
\hline
V^x_{01} & & & & & & V^q_{01} & & \\
 & V^x_{02} & & & & & & V^q_{02} & \\
 & & & & & V^q_{10} & & & V^q_{10} \\
 & & V^x_{11} & & & & V^q_{11} & & \\
 & & & V^x_{12} & & & & V^q_{12} & \\
\hline
R^x_{01} & R^x_{02} & & & R^x_{00} & R^q_0 & R^q_{01} & R^q_{02} & \\
 & & R^q_{11} & R^q_{12} & & R^q_{10} & R^q_{11} & R^q_{12} & R^x_{10} \\
\tilde{R}^x_{01} & \tilde{R}^x_{02} & \tilde{R}^x_{11} & \tilde{R}^x_{12} & \tilde{R}^x_{00} & \tilde{R}^q_{00}+\tilde{R}^q_{10} & \tilde{R}^q_{01}+\tilde{R}^q_{11} & \tilde{R}^q_{02}+\tilde{R}^q_{12} & \tilde{R}^x_{10}
\end{array}\right)$$

Im allgemeinen Fall ist die Struktur der Jacobi-Matrix dann wie in Abbildung 3.3 dargestellt. Hierbei sind alle Indices weggelassen und $*$ bezeichnet das “Fill-in”, das durch die Vorkondensierung entstanden ist.
Analog zum Single-Setpoint-Fall kann man nun eine blockweise Gauß-Elimination durchführen und eine Transformationsmatrix T finden, sodaß

$$T\left(\begin{array}{c} CP^T \\ \hline DP^T \end{array}\right) = \left(\begin{array}{c|c} -I & C'_2 \\ \hline 0 & D'_2 \end{array}\right)$$

und kann wie in Abschnitt 2.3.4 beschrieben das kondensierte QP herleiten.

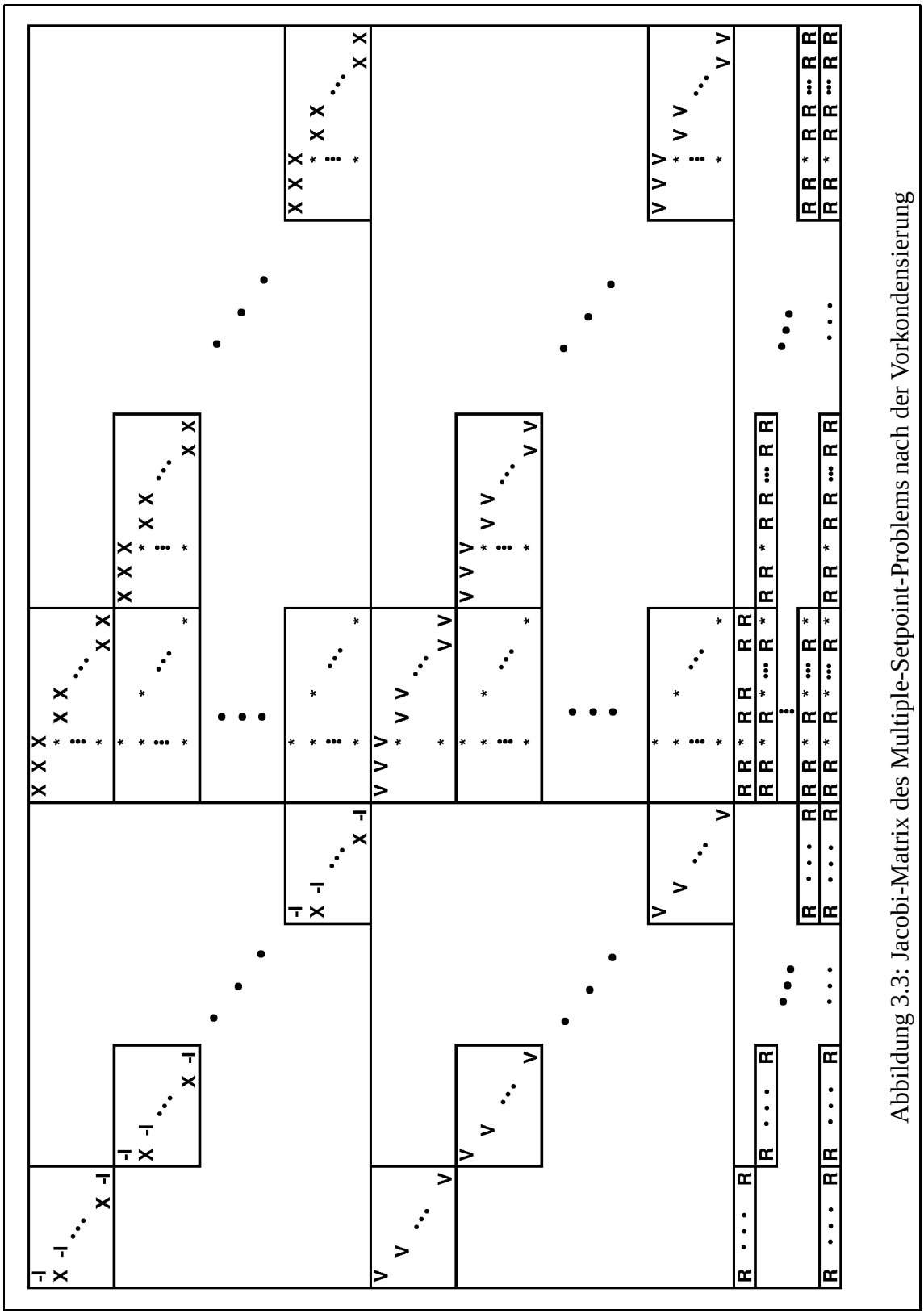

Abbildung 3.3: Jacobi-Matrix des Multiple-Setpoint-Problems nach der Vorkondensierung

3.6 SQP-Algorithmus für Multiple-Setpoint-Probleme

In diesem Abschnitt ist ein vollständiger SQP-Algorithmus für Multiple-Setpoint-Probleme der optimalen Steuerung auf Basis der Mehrziel-Methode (Abschnitt 1.3) aufgeführt.

ALGORITHMUS 3.1 (SQP-VERFAHREN)

1. Beginne mit $k := 0$ und einer Startlösung w_0. Die Startlösung kann vorgegeben werden oder durch Integration (Single-Shooting) erzeugt werden.

2. Berechne zu allen Setpoints Zielfunktion, Nebenbedingungen und alle Ableitungen:
   $F_i(w_0)$, $x_i(w_0)$, $g_i(w_0)$, $h_i(w_0)$, $r_{ij}(w_0)$, $\tilde{r}_{ij}(w_0)$
   $X_{ij}^x(w_0)$, $X_{ij}^z(w_0)$, $X_{ij}^{\hat{q}}(w_0)$,
   $V_{ij}^x(w_0)$, $V_{ij}^z(w_0)$, $V_{ij}^{\hat{q}}(w_0)$,
   $R_{ij}^x(w_0)$, $R_{ij}^z(w_0)$, $R_{ij}^{\hat{q}}(w_0)$,
   $\tilde{R}_{ij}^x(w_0)$, $\tilde{R}_{ij}^z(w_0)$, $\tilde{R}_{ij}^{\hat{q}}(w_0)$
   Die meisten Berechnungen können separat ausgeführt werden, sodaß sich der Algorithmus zur Parallelisierung eignet.
   Die bei der Berechnung von $X_{ij}^x(w_0)$, $X_{ij}^z(w_0)$, $X_{ij}^{\hat{q}}(w_0)$ auftretenden Anfangswertprobleme werden mit einem geeigneten DAE-Löser, z.B. einem Adams-Bashforth-Moulton-Verfahren (Abschnitt 1.1.3) gelöst. Dabei werden mithilfe von interner numerischer Differentiation (Abschnitt 2.3.1). die Ableitungen $X_{ij}^x(w_0)$, $X_{ij}^z(w_0)$, $X_{ij}^{\hat{q}}(w_0)$ berechnet.

3. Wähle eine Anfangsnäherung für die Hesse-Matrix B_0 nach Plitt (Abschnitt 2.2.1.1)

4. Formuliere das aktuelle QP

$$\begin{aligned} \min_{\Delta w_k} \frac{1}{2}\Delta w_k^T B_k \Delta w + \nabla F(w_k)^T \Delta w_k \\ G(w_k) + \nabla G(w_k)\Delta w_k &= 0 \\ H(w_k) + \nabla H(w_k)\Delta w_k &\geq 0 \end{aligned}$$

5. Führe Vorkondensierung und Kondensierung (Abschnitt 2.3.4) durch, um das kondensierte QP zu erhalten.

6. Löse das kondensierte QP mithilfe der Active-Set-Strategie (Abschnitt 2.2.3.3), berechne die Lösung des ursprünglichen QPs durch Rekursionsformeln (Abschnitt 2.3.4) und erhalte Δw_k sowie die Lagrange-Multiplikatoren $\tilde{\lambda}_k, \tilde{\mu}_k$.

7. Konvergenzcheck: Falls

$$\|\nabla F(w_k)^T \Delta w_k\| + \sum_{i=1}^{m} \mid (\tilde{\lambda}_k)_i G_i(w_k) \mid + \sum_{i=1}^{l} \mid (\tilde{\mu}_k)_i H_i(w_k) \mid \quad \leq \quad \varepsilon_{Tol}$$

   erfüllt ist, beende Algorithmus mit $w_k, \tilde{\lambda}_k, \tilde{\mu}_k$ als Lösung.

6. Führe eine Armijo-Goldstein-Liniensuche (Abschnitt 2.2.4.1) basierend auf der Watchdog-Technik (Abschnitt 2.2.4.3) durch und erhalte t_k aus einer Näherungslösung von

$$\min_{t_k} T(w_k + t_k \Delta w_k)$$

   mit geeigneter Gütefunktion T (Abschnitt 2.2.4.2).

8. Führe die Iteration durch

$$\begin{aligned} w_{k+1} &:= w_k + t_k \Delta w_k \\ \lambda_{k+1} &:= \lambda_k + t_k(\tilde{\lambda}_k - \lambda_k) \text{ (oder } \lambda_{k+1} := \tilde{\lambda}_k) \\ \mu_{k+1} &:= \mu_k + t_k(\tilde{\mu}_k - \mu_k) \text{ (oder } \mu_{k+1} := \tilde{\mu}_k) \end{aligned}$$

9. Berechne "alten" Gradienten mit neuen Multiplikatoren

$$\nabla_w \mathcal{L}(w_k, \lambda_{k+1}, \mu_{k+1}) \quad = \quad \nabla F(w_k) - \nabla G(w_k)\lambda_{k+1} - \nabla H(w_k)\mu_{k+1}$$

   (Leicht zu berechnen)

10. Berechne Funktionen und Ableitungen am neuen Punkt w_{k+1}:
    Berechne zu allen Setpoints Zielfunktion, Nebenbedingungen und alle Ableitungen am neuen Punkt w_{k+1}:
    $F_i(w_{k+1})$, $x_i(w_{k+1})$, $g_i(w_{k+1})$, $h_i(w_{k+1})$, $r_{ij}(w_{k+1})$, $\tilde{r}_{ij}(w_{k+1})$
    $X_{ij}^x(w_{k+1})$, $X_{ij}^z(w_{k+1})$, $X_{ij}^{\hat{q}}(w_{k+1})$,
    $V_{ij}^x(w_{k+1})$, $V_{ij}^z(w_{k+1})$, $V_{ij}^{\hat{q}}(w_{k+1})$,
    $R_{ij}^x(w_{k+1})$, $R_{ij}^z(w_{k+1})$, $R_{ij}^{\hat{q}}(w_{k+1})$,
    $\tilde{R}_{ij}^x(w_{k+1})$, $\tilde{R}_{ij}^z(w_{k+1})$, $\tilde{R}_{ij}^{\hat{q}}(w_{k+1})$
    (Einzelheiten wie bei Schritt 2.)

11. Berechne “neuen” Gradienten

$$\nabla_w \mathcal{L}(w_{k+1},\lambda_{k+1},\mu_{k+1}) \;=\; \nabla F(w_{k+1}) - \nabla G(w_{k+1})\lambda_{k+1} - \nabla H(w_{k+1})\mu_{k+1}$$

12. Berechne blockweise das Update der Hesse-Matrix (Abschnitt 2.3.3)

$$B_{k+1} \;=\; B_k + \sum_{i=0}^{N-1} \sum_{j=0}^{\hat{m}_i} U_{ij}(B_k,p_k,q_k)$$

aus

$$\begin{aligned} p_k &= w_{k+1} - w_k \\ q_k &= \nabla_w \mathcal{L}(w_{k+1},\lambda_{k+1},\mu_{k+1}) - \nabla_w \mathcal{L}(w_k,\lambda_{k+1},\mu_{k+1}) \end{aligned}$$

Dabei bezeiche $U(B_k, p, q)$ eine geeignete Update-Formel, z.B. BFGS-Update (siehe Abschnitt 2.2.1.2).

13. $k := k + 1$ und gehe zu Schritt 5.

4 Anwendung: Multiple-Setpoint-Optimierung bei der Komfort-Optimierung in Automobilen

Als Anwendungsbeispiel für den neuen Multiple-Setpoint-Algorithmus wollen wir ein mechanisches Modell eines Automobils in Form eines Mehrkörpersystems betrachten, welches von der Firma Freudenberg bereitgestellt wurde. In einem gemeinsamen Projekt mit Freudenberg sollte die Übertragung von Schwingungen von Straße und Motor auf die Fahrgastzelle untersucht werden. Das Ziel des Projekt war es, einzelne Komponenten des Modells, und zwar die Motorlagerung, zu optimieren. Die Optimierung sollte durch eine minimale Schwingungsübertragung im gesamten Frequenzbereich den Fahrkomfort verbessern.
Im ersten Abschnitt werden wir das gegebene Problem näher erläutern. Der zweite Abschnitt beinhaltet allgemeine Dinge über die Behandlung von Mehrkörpersystemen. Im dritten Abschnitt werden wir die numerischen Resultate der Simulation des Automodells und der Optimierung desselben mithilfe des im vorangegangenen Kapitels Multiple-Setpoint-Algorithmus des Automodells präsentieren. [1]

4.1 Problemstellung

Das hier betrachtete Automodell besteht aus sechs Körpern: Vier Rädern, Karosserie und Motor. Vernachlässigt sind weitere Körper wie Tilger, Auspuff usw. Verbunden sind die Körper durch insgesamt 25 lineare Feder-Dämpfer-Elemente. Die Räder sind mit dem Inertialsystem, das ist in unserem Fall die Straße, verbunden durch je eine 3-dimensionale Feder, modelliert durch je drei 1-dimensionale Federn. Außerdem sind die Räder mit der Karosserie durch ein prismatisches Gelenk und einem Feder-Dämpfer-Element verbunden. Der Motor ist quer eingebaut und mit der Karosserie verbunden durch zwei Lager seitlich im vorderen, oberen Bereich und einer Pendelstütze in der Mitte an der hinteren unteren Kante.
Anregungen können im Automodell an zwei Stellen eingebracht werden. Erstens an der Vorderachse, und zwar durch eine sinusförmige Anregung der Vorderräder mit wählbaren Amplitude

[1] Anmerkung: An den meisten Diagrammen in diesem Abschnitt wurden aufgrund einer Geheimhaltungsvereinbarung mit der Firma Freudenberg die Beschriftung der y-Achse entfernt.

und Frequenz, die Bodenwellen simulieren sollen. Zweitens am Motor durch eine Drehschwingung um die y-Achse mit wählbarer Amplitude und Frequenz. Im konkreten Fall sollten Anregungen mit einer Frequenz von $2 - 20$ Hz berücksichtigt werden mit einer Amplitude von 1mm (Radanregung) beziehungsweise 100 Nm (Motoranregung) um die y-Achse.
Optimiert werden sollten entweder die Positionen der Lager oder die Steifigkeiten in Hinblick auf eine Minimierung der Schwingungsübertragung zu einem Punkt auf der Karosserie, an der in der Praxis der Fahrersitz montiert wird. Bei den Positionen war eine maximale Verschiebung der Lager um ± 30 Millimeter in die drei Raumrichtungen erlaubt, bei den Steifigkeiten war der Wertebereich eingeschränkt auf $50 - 500$ N/mm .
Als zusätzliche Nebenbedingung war gefordert, daß der maximale Verdrehwinkel des Motors beschränkt ist ($\|\beta_x\| \leq 0.5°, \|\beta_y\| \leq 0.5°, \|\beta_z\| \leq 2.5°$).
Abbildung 4.1 zeigt eine schematische Darstellung des Automodells.

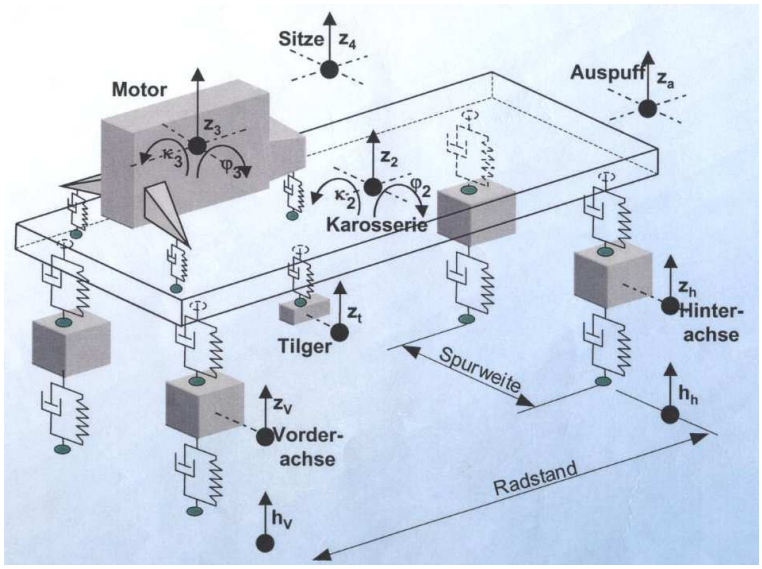

Abbildung 4.1: Automodell

4.2 Allgemeines über Mehrkörpersysteme

In den folgenden Abschnitten werden wir das verwendete mechanische Modell erklären. Nach einer kurzen Einführung in Mehrkörpersysteme werden wir das Prinzip der natürlichen Koordinaten einführen und deutlich machen, warum diese Art der Darstellung von mechanischen Modellen für eine numerische Simulation besonders geeignet ist. Es folgt eine eine Formulierung der mechanischen Bewegungsgleichungen und eine Beschreibung des verwendeten Softwarepakets MBSNAT.

4.2.1 Mehrkörpersysteme

Die computergestützte Simulation und Optimierung von Mehrkörpersystemen spielen heutzutage eine wichtige Rolle in der technischen Mechanik. Nachdem lange Zeit keine geeigneten Werkzeuge bereit standen, die eine schnelle und genaue Simulation der Mehrkörpersysteme ermöglichten, wurden in den letzten Jahren schnelle und strukturausnutzende Verfahren entwickelt. Numerische Simulation und Optimierung von Mehrkörpersystemen wurden zur Schlüsseltechnologie. Eine Vielzahl von Anwendungsgebieten ergeben sich in der Fahrzeugdynamik, Robotik, Biomechanik und in zahlreichen Bereichen der Ingenieurwissenschaften.
Die Formulierung eines Mehrkörpersystems liefert eine pysikalische Beschreibung eines Systems von starren Körpern, die verbunden bzw. gekoppelt sind durch Gelenke und Kraft-Elemente.
Die einzelnen Körper werden als starr, das heißt komplett inelastisch betrachtet. Sie können als Ganzes rotieren oder sich verschieben, aber ändern nicht ihre Form. Sie werden beschrieben durch ihre Position und Ausrichtung im Raum (sechs Freiheitsgrade) sowie ihre Masse und ihr Trägheitsmoment.
Gelenke verbinden jeweils zwei Körper und schränken die Freiheitsgrade der relativen Bewegung der beiden Körper zueinander ein. Typische Gelenke sind prismatische Gelenke, Drehgelenke, Kugelgelenke, etc.
Kraftelemente wie Feder- und Dämpferelemente beschreiben die Kraftübertragung zwischen zwei Körpern. Die Kraftelemente werden in der Regel als masselos angenommen. Die Kräfte, die sie verursachen, hängen von der relativen Position und der relativen Geschwindigkeit der beiden entsprechenden Körper ab. Zusätzlich können Kräfte von außen eingebracht werden. Auch die Gravitation wird als externe Kraft behandelt. Typischerweise wird angenommen, daß die Kräfte und Momente an einem Punkt des Körpers beziehungsweise an einer Achse des Körpers angreifen.
Auf den ersten Blick mag dies eine große Idealisierung der Realität sein, aber für viele Anwendungen aus Bereichen der Technik oder der Biomechanik ([Kra05]) ist dieses Modell eine geeignete Approximation.

4.2.2 Wahl des Koordinatensystems / Natürliche Koordinaten

Ein wichtiges Augenmerk liegt beim Modellieren von Mehrkörpersystemen auf der Wahl eines geeigneten Koordinatensystems. Kinematische Systeme mit Baum-Struktur und ohne geschlossene Schleifen können in Minimalkoordinaten beschrieben werden. Dafür gibt es verschiedene Techniken, zum Beispiel rekursive, entlang der kinematischen Kette operierende Verfahren, die Verwendung von Lagrange-Gleichungen zweiter Art oder Newton-Euler-Methoden. Dies führt zu Bewegungsgleichungen in Form eines Systems gewöhnlicher Differentialgleichungen. Systeme mit kinematischen Schleifen sind mit diesem Ansatz wesentlich schwieriger zu behandeln.
Eine andere Vorgehensweise basiert auf *redundanten Koordinatensystemen* auf der Basis von Lagrange-Gleichungen der ersten Art. Dies führt zu einem System differentiell-algebraischer Gleichungen vom Index 3. Bei den meisten Modellierungstechniken für Mehrkörpersysteme

wird jedem Körper des Systems ein lokales, körpereigenes Koordinatensystem zugeordnet. Die Wahl der Koordinaten beschreibt meist eine Parametrisierung der affinen Transformation zwischen globalen Inertialsystem und körpereigenem Koordinatensystem. *Referenzpunkt-Koordinaten* zum Beispiel benutzen die kartesischen Koordinaten des Ursprungs des körpereigenen Koordinatensystems sowie die drei Eulerwinkel, die die Verdrehung des lokalen Koordinatensystem beschreiben. Diese Darstellung ist zwar sehr kompakt, weil sie mit sechs Koordinaten pro Körper auskommt, weist aber zwei singuläre Positionen auf, was numerisch für große Schwierigkeiten sorgen kann. Man kann zeigen, daß jede Parametrisierung mit sechs Koordinaten mindestens zwei singuläre Punkte besitzt. Um diesen Nachteil zu umgehen, muß man daher auf redundante Koordinatensysteme mit mehr als sechs Koordinaten pro Körper zurückgreifen.
Wir verwenden eine Variante der *natürlichen Koordinaten*, die von Kraus ([Kra05], [Kra97]) vorgeschlagen wurde und auf Methoden von García de Jalón und Bayo ([GdJB94]) aufbaut. Die Methode verwendet für jeden Körper zwölf Koordinaten, führt aber zu sehr einfachen, strukturierten Gleichungen, typischerweise von maximal zweiter Ordnung. Das Differentialgleichungssytem ist demnach von hoher Dimension, aber die Einträge sind sehr einfach zu berechnen. Jalón und Bayo nutzen die entstehende Struktur, um Variablen von verschiedenen durch Gelenken verbundenen Körpern zusammenzufassen, um die Dimension des Systems zu reduzieren. Bei der Methode von Kraus wird auf diese Zusammenfassungen verzichtet, um die Struktur der System-Matrix zu erhalten und weiter ausnutzen zu können. Dadurch ist eine Auswertung der rechten Seite der Differentialgleichung in $\mathcal{O}(n)$ möglich – gegenüber $\mathcal{O}(n^2)$ bei der Methode von Jalón und Bayo –, wobei n die Anzahl der Körper ist. Um die Methode anwenden zu können, muß gewährleistet sein, daß das System keine redundanten Nebenbedingungen enthält, was durch eine besondere Formulierung der Nebenbedingungen erreicht wird.
Jeder Körper erhält sein eigenes, körperfestes Koordinatensystem und wird beschrieben durch die drei Koordinaten seines Schwerpunkts und drei dreidimensionale Vektoren, die die Koordinatenachsen des körpereigenen Koordinatensystem darstellen, also durch insgesamt zwölf Koordinaten. Sei r ein körperfester Punkt ausgedrückt in lokalen Koordinaten, dann ergibt sich seine Position im globalen Koordinatensystem $\bar{r}$ durch eine affine Transformation:

$$
\begin{aligned}
\bar{r} &= o + Xr && \text{(4.1a)}\\
&= 1o + (e_1|e_2|e_3)r && \text{(4.1b)}\\
&= 1o + r_1e_1 + r_2e_2 + r_3e_3 && \text{(4.1c)}\\
&= \underbrace{\left(\begin{array}{ccc|ccc|ccc} 1 & & & r_1 & & & r_2 & & & r_3 & & \\ & 1 & & & r_1 & & & r_2 & & & r_3 & \\ & & 1 & & & r_1 & & & r_2 & & & r_3 \end{array}\right)}_{=:C_r} \underbrace{\begin{pmatrix} o \\ e_1 \\ e_2 \\ e_3 \end{pmatrix}}_{=:y} && \text{(4.1d)}\\
&= C_r y && \text{(4.1e)}
\end{aligned}
$$

Dabei ist $o \in \mathbb{R}^3$ die Verschiebung des Ursprungs und $X \in \mathbb{R}^{3\times3}$ die Orientierungsmatrix (siehe Abbildung 4.2). $C_r \in \mathbb{R}^{3\times12}$ ist bestimmt durch die lokalen affinen Koeffizienten und $y \in \mathbb{R}^{12}$ sind die dynamischen Variablen des Körpers.

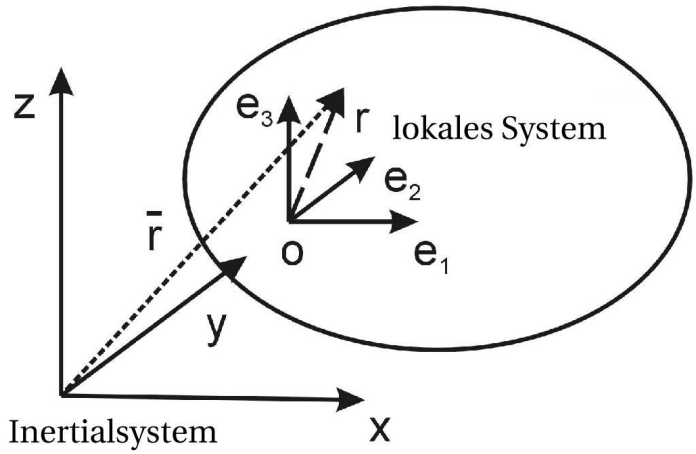

Abbildung 4.2: Affine Transformation bei natürlichen Koordinaten

Basierend auf dieser Transformation kann eine einfache Darstellung der Massenmatrix hergeleitet werden. Aufgrund des körperfesten Koordinatensystems bleibt die Massenmatrix zeitlich konstant; es treten keine Coriolis- oder Zentrifugalkräfte im Modell auf.

Durch zusätzliche Nebenbedingungen (Rechtwinkligkeit des Koordinatensystems, Normiertheit der Achsen) wird die Anzahl der Freiheitsgrade (sechs Freiheitsgrade) eines Einzelkörpers erhalten.

Gelenke schränken die Freiheitsgrade des kompletten Systems weiter ein. Sie werden modelliert durch zusätzliche Nebenbedingungen. Im allgemeinen sind dies Beschränkungen an eine translatorische Bewegung in eine Richtung oder Rotationsbewegungen um eine Achse. Durch die Linearität der affinen Transformation (4.1e) sind die entstehenden Gleichungen höchstens quadratisch bezüglich der dynamischen Variablen.

Die Wahl der natürlichen Koordinaten bewirkt also neben der zeitlich konstanten Massematrix eine Reduzierung der Nichtlinearität der Nebenbedingungen (Invarianten) und des Krümmungsterms γ.

4.2.3 Mechanisches DAE-Modell

Mechanische Systeme können beschrieben werden als DAE vom Index 3

$$\begin{aligned} M(y)\ddot{y} &= f(t,y,\dot{y}) - G(y)^T\lambda \\ g(y) &= 0 \end{aligned}$$

wobei folgende Bezeichnungen gelten:

y		(generalisierte) Positionen
$M(y)$		Massenmatrix
$f(t,y,\dot{y})$		explizite Kraft-Terme
$g(y)$		kinematische Nebenbedingungen
$G(y)$	$:=$	$\frac{\partial}{\partial y} g(y)$ Jacobi-Matrix der Nebenbedingungen
λ		Lagrange-Multiplikatoren

Die kinematischen Nebenbedingungen werden als scleronom und holonom, das heißt unabhängig von t und $\dot{y}$ angenommen.

Index-Reduktion führt zu einer quasi-linearen DAE vom Index 1.

$$\dot{y} = v \tag{4.2a}$$

$$\dot{v} = a \tag{4.2b}$$

$$\begin{pmatrix} M(y) & G(y)^T \\ G(y) & 0 \end{pmatrix} \begin{pmatrix} a \\ \lambda \end{pmatrix} = \begin{pmatrix} f(t,y,v) \\ \gamma(t,y,v) \end{pmatrix} \tag{4.2c}$$

wobei

$$\gamma(y,v) := G(y)a - \frac{d^2}{dt} g(y)$$

In dieser Schreibweise können die Positionen y und Geschwindigkeiten v als differentielle Variablen aufgefaßt werden, die Lagrange-Multiplikatoren λ und die Beschleunigungen a als algebraische Variablen.

Die Bedingung $g(y) = 0$ ist vom Index 3 und ist implizit enthalten in der Beschleunigungs-Nebenbedingung vom Index 1 in (4.2c). Die ursprüngliche Gleichung $g(y) = 0$ und seine Ableitung $G(y)v = 0$ werden als Positionsinvariante und Geschwindigkeitsinvariante bezeichnet. Sie werden dazu benutzt, auf die durch die Invarianten definierten Mannigfaltigkeit zu projizieren, um einen Drift der numerischen Lösung weg von der Mannigfaltigkeit zu verhindern ([Sha86], [Eic91]). Weiterhin wollen wir das System erweitern um *externe Variablen* e mit externer Dynamik $\breve{f}$, das heißt dynamische Variablen, die anderen, nicht-mechanischen Differentialgleichungen genügen und das mechanische System "von außen" beeinflussen. Beispiele sind hydraulische oder elektronische Elemente oder von außen eingebrachte Steuerungen.

Unter Einbeziehung der Invarianten und der externen Dynamik erhalten wir die *volle Deskriptorform*:

$$
\begin{aligned}
\dot{e} &= \breve{f}t,y,v,e) \\
\dot{y} &= v \\
\dot{v} &= a \\
\begin{pmatrix} M(y,e) & G(y,e)^T \\ G(y,e) & 0 \end{pmatrix} \begin{pmatrix} a \\ \lambda \end{pmatrix} &= \begin{pmatrix} f(t,y,v,e) \\ \gamma(t,y,v,e) \end{pmatrix} \\
g(y) &= 0 \quad \text{Positions-Invariante} \\
G(y)v &= 0 \quad \text{Geschwindigkeits-Invariante}
\end{aligned}
$$

Damit das System lösbar wird, stellen wir folgende weiteren Voraussetzungen: G habe vollen Rang und M sei positiv definit auf dem Kern von G.

4.2.4 Softwarepakete MBSNAT / MBSSIM

Das mathematische Modell für die Bewegungsgleichungen wurde mit dem am Interdisziplinären Zentrum für Wissenschaftliches Rechen der Universität Heidelberg (IWR) entwickelten Tool MBSNAT ([Kra05]) entwickelt. MBSNAT ist ein Werkzeug zur Modellierung von mechanischen Systemen in natürlichen Koordinaten. Die Eingabe des Modells erfolgt bequem in Form von xml-Files. [2]
Starrkörper werden beschrieben durch Ausmaße, Anfangspositionen, Massen und Trägheitsmomente. Hier die Formulierung des Motors im verwendeten xml-Dateiformat.

```
<body>
        <id> Motor </id>
        <position>
                <o> -0.1823 -0.0355 0.2057 </o>
                <x>  0.0  1.0  0.0 </x>
                <y> -1.0  0.0  0.0 </y>
                <z>  0.0  0.0  1.0 </z>
        </position>
        <mass> 120.4 </mass>
        <center_of_mass> 0.0 0.0 0.0 </center_of_mass>
        <moment_of_inertia>
                <x>  5.78 -0.45 -1.75 </x>
                <y> -0.67  7.56 -0.26 </y>
                <z> -1.60 -0.32  4.21 </z>
        </moment_of_inertia>
</body>
```

Gelenke werden beschrieben durch zwei Punkte auf zwei verschiedenen Körpern, die verbunden werden sollen, und der Art des Gelenks.
Folgende Gelenke sind implementiert:

[2] Anmerkung: Die Datenwerte in diesem Abschnitt sind nicht original und daher beispielhaft.

- Kugelgelenk (spherical joint)
- Drehgelenk (revolute joint)
- Kardangelenk (universal joint)
- Zylindergelenk (revolute joint)
- Schubgelenk (prismatic joint).

Hier die Formulierung ein Schubgelenks:

```
<joint key="Prismatic">
        <id> Radaufhaengung_vorne_links </id>
        <end id="Karosserie">
                <o> -1.3789 -0.7195 -0.3124 </o>
                <x> 1.0 0.0 0.0 </x>
                <y> 0.0 1.0 0.0 </y>
                <z> 0.0 0.0 1.0 </z>
        </end>
        <begin id="Linkes_Vorderrad">
                <o> 0.0 0.0 0.0 </o>
                <x> 1.0 0.0 0.0 </x>
                <y> 0.0 1.0 0.0 </y>
                <z> 0.0 0.0 1.0 </z>
        </begin>
</joint>
```

Kraftelemente werden beschrieben durch zwei Punkte auf zwei verschiedenen Körpern, die die Richtung der Kraft definieren, einem “Key”, der angibt, um was für ein Kraftelement es sich handelt, sowie je nach Art des Kraftelements einiger Konstanten.
Folgende Kräfte sind implementiert:

- *externe Kräfte:* Gravitation (Zweiter Körper ist hier das Inertialsystem)
- *translatorische Kräfte*: Die Kräfte sind dadurch definiert, daß sie in Richtung der Verbindungslinie beider Punkte wirken und nur die Projekton verschiedener Größen auf diese Richtung verwendet wird (Spring, Damper, Actuator).
- *Z-Force:* Hierbei handelt es sich um idealisierte Kräfte, die in eine globale Richtung zeigen unabhängig vom Zustand des Systems (Z-Spring, Z-Damper, Z-Actuator).
- *Rotatorial:* Hier geht der Winkel zwischen zwei Vektoren senkrecht zu einer Drehachse in die Berechnung ein und verursacht eine rotatorische Kraft um diese Achse (Rotatorial Spring, Rotatorial Damper, Rotatorial Actuator).
- *Sinusanregungen (im Rahmen dieser Arbeit neu implementiert):* Sinusförmige translatorische oder rotatorische Anregung von einzelnen Körpern.

Hier die Formulierung eines Feder-Dämpfer-Elements. Die Konstanten legen die Ruhelänge, Steifigkeit und Dämpfung fest.

```
<force key="Direction-Spring_Damper">
        <id> Motorfeder_x1 </id>
        <end id="Motor">
                <o> 0.4743 0.0207  0.1403 </o>
                <x> 1.0 0.0 0.0 </x>
                <y> 0.0 1.0 0.0 </y>
                <z> 0.0 0.0 1.0 </z>
        </end>
        <begin id="Karosserie">
                <o> -1.4300 0.4500 0.0090 </o>
                <x> 0.0 0.0 1.0 </x>
                <y> 0.0 1.0 0.0 </y>
                <z> 1.0 0.0 0.0 </z>
        </begin>
        <constants>
        0.0 330000.0 100.0
        </constants>
</force>
```

Die von MBSNAT erzeugten Modellgleichungen werden mit dem ebenfalls am IWR entwickelten Integrator MBSSIM gelöst, der speziell auf mechanische Systeme zugeschnitten ist ([vSW94], [vS99]). MBSSIM ist ein schneller und zuverlässiger Integrator von Mehrkörpersystemen in Deskriptorform. Zur schnellen und direkten Behandlung von linear-impliziten Systemen in Zustandsform (mechanische ODE) oder Deskriptorform (mechanische DAE vom Index 3) bietet MBSSIM verschiedene Diskretisierungsmethoden (Adams-Bashforth-Moulton-Verfahren, Runge-Kutta-Verfahren und Extrapolations-Verfahren). Wir verwenden das in MBSSIM implementierte Adams-Bashforth-Moulton-Verfahren, das sich für das Automodell besonders gut eignet. (Vergleiche Abschnitt 1.1.3)

4.3 Optimierung des Automodells

Zunächst werden wir die Ergebnisse der Simulation betrachten und das Problem analysieren. Im Folgenden wird dann eine geeignete Zielfunktion formuliert, bevor wir schließlich die Ergebnisse der Optimierung vorstellen, die der neue Multiple-Setpoint-Algorithmus produzierte. Alle Rechenzeiten beziehen sich auf Rechnungen auf einem PC mit einem Pentium 4 Prozessor mit 2.53 GHz.

4.3.1 Ergebnisse der Simulation

Der erste Schritt bei der Simulation ist die Berechnung der eingefederten Ruhelage, die fortan als Anfangswert für die Simulationen dient. Aus der Ruhelage heraus wird das Modell mit der

gewählten Anregung in Bewegung gesetzt, und es werden die dynamischen Bewegungsgleichungen gelöst. Es zeigt sich, daß sich nach einer Einschwingphase ein stationärer Zustand in Form einer periodischen Bewegung einstellt. Man unterteilt die Bewegung also in eine Einschwingphase und eine Meßphase.

Wir unterscheiden zwei Arten von Einschwingvorgängen. Erstens das durch die Gravitation verursachte Einschwingen in das Gleichgewicht zwischen Federkräften und Gravitation, das auch ohne Eingabe von äußeren Anregungen geschieht. Zweitens das Einschwingen in eine periodische Bewegung entsprechend der anregenden Schwingung. Beide Einschwingvorgänge überlagern sich zu Beginn der Simulation.
Abbildung 4.3 zeigt den Einschwingvorgang, wenn man die Anregung und die Simulation in einem Zustand startet, in dem schon ein Gleichgewicht zwischen Federkräften und Gravitation herrscht.

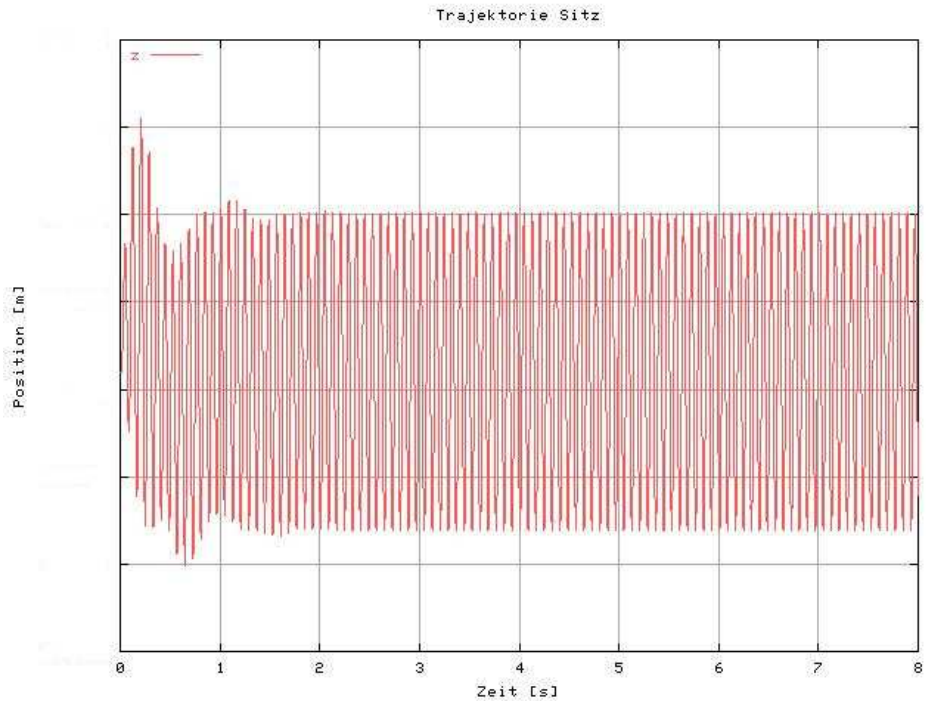

Abbildung 4.3: Einschwingung aus eingefederter Ruhelage in angeregte Schwingung

Der stationäre eingeschwungene Zustand ändert sich jedoch im Laufe der Optimierung, da gewisse Komponenten des Modells (Positionen oder Steifigkeiten der Lager) verändert werden. Die gegebenen Anfangswerte beschreiben dann nicht mehr das Gleichgewicht von Federkräften und Gravitation. Abbildung 4.4 zeigt den Einschwingvorgang in eine solche modifizierte Ruhelage.
Im Laufe der Optimierung haben wir eine Überlagerung dieser beiden Einschwingvorgänge. Dies ist in Abbildung 4.5 dargestellt.

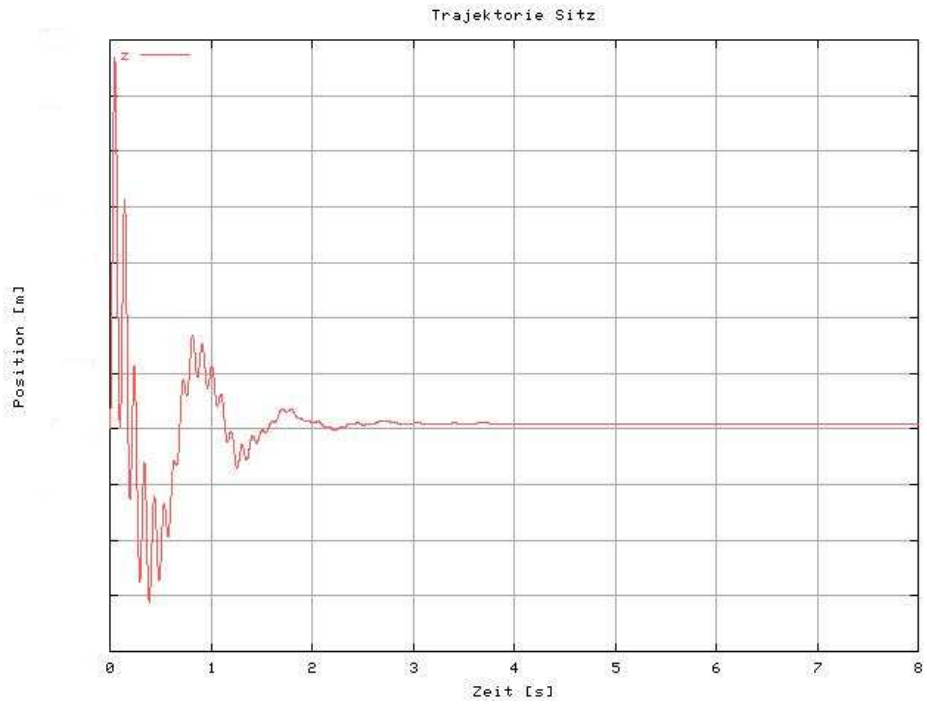

Abbildung 4.4: Einschwingung in geänderte Ruhelage

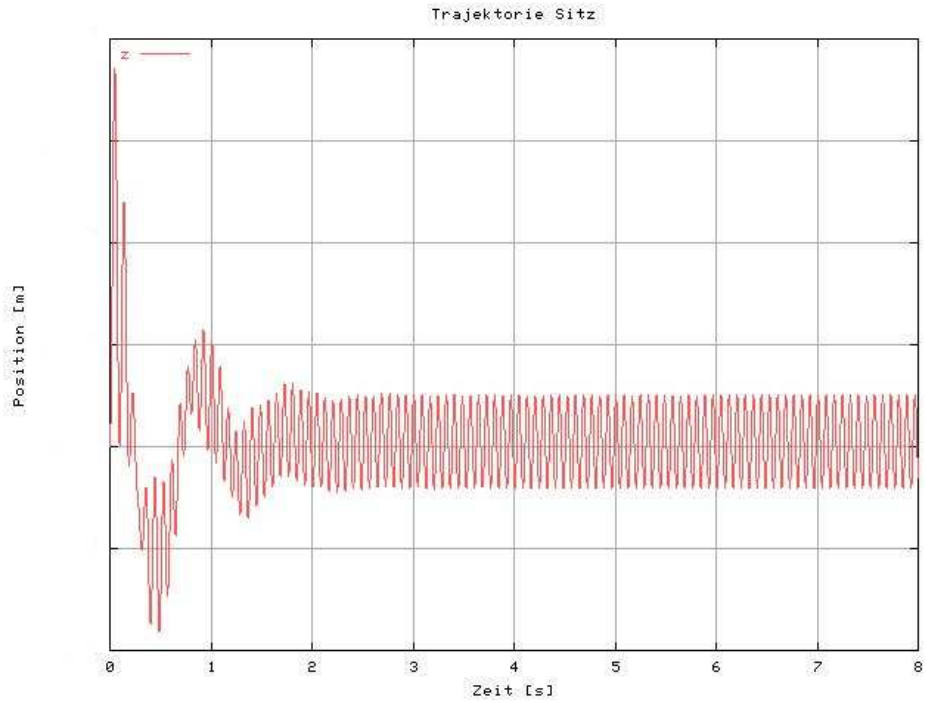

Abbildung 4.5: Überlagerung der beiden Einschwingvorgänge

Nach einer hinreichend langen Zeitdauer ist die Einschwingphase abgeschlossen und die übertragene Schwingung kann in der Meßphase mit einer Fourier-Analyse untersucht werden. Um eventuell noch von der Einschwingphase herrührende nicht-periodische Anteile herauszufiltern wird über einen größeren Zeitraum als eine Periode integriert.
Eine Kette von Simulationen, ein sogenannter “Sweep”, ergibt Resonanzkurven für das Modell. Die Fourier-Analyse zeigt, dass bei der Radanregung nur die Grundschwingung übertragen wird, wie man es bei einem rein linearen System auch erwartet. Bei der Motoranregung durch eine Drehschwingung zeigt sich, daß bei bestimmten Frequenzen auch die erste Oberschwingung auftritt.
Die Abbildungen 4.6-4.9 zeigen die Frequenzanalyse der Beschleunigungen am Fahrersitz bei Anregung des Motors mit variabler Frequenz. Man kann sehen, daß sich die auf den Fahrersitz übertragene Bewegung zusammensetzt aus einer Grundschwingung mit der Frequenz der Anregung (Hauptdiagonale) und einer ersten Oberschwingung mit geringerer Amplitude (Nebendiagonale). Es genügt also, im folgenden die Fourier-Analyse nur für die Grundschwingung und die erste Oberschwingung durchzuführen.
Die erste Abbildung zeigt die Frequenzanalyse des Absolutbetrags der Beschleunigungen, die folgenden Abbildungen die Frequenzanalysen der Beschleunigungen in x-, y- und z- Richtung.

Die Ergebnisse der Fourier-Analyse für die Grundschwingung können verglichen werden mit den von Freudenberg berechneten Werten, die mit einem linearen Modell und einer Berechnung im Frequenzraum erzeugt wurden. Gerechnet wurde jeweils ein Sweep von 2-22 Hz in 0.1 Hz Schritten (200 Simulationen) für Radanregung und Motoranregung. Die Rechenzeit für einen Sweep beträgt ca. 40 Minuten.
Abbildung 4.10 zeigt die Beschleunigungen am Fahrersitz bei Anregung der Vorderachse mit 1 mm Amplitude in den drei Raumrichtungen im Frequenzbereich $2-25$ Hz im Vergleich zu den Berechnungen von Freudenberg (gestrichelt). Man sieht die unterschiedlichen Resonanzen in den verschiedenen Raumrichtungen. Die größte Schwingungsübertragung findet im Frequenzbereich von $10-20$ Hz statt, das Maximum liegt knapp über 10 Hz. Trotz des allgemeineren Ansatzes stimmen die Berechnungen gut mit den Werten von Freudenberg überein.
Abbildung 4.11-4.13 zeigt die Beschleunigungen bei Anregung der Vorderachse mit 1 mm Amplitude in den drei Motorlagern in x-, y- und z-Richtung im Vergleich zu den Berechnungen von Freudenberg (gestrichelt).
Abbildung 4.14 zeigt die Beschleunigungen am Fahrersitz bei Anregung des Motors durch eine Drehschwingung um die y-Achse mit einer Amplitude von 100 Nm in den drei Raumrichtungen im Frequenzbereich $2-25$ Hz im Vergleich zu den Berechnungen von Freudenberg (gestrichelt). Die größte Schwingungsübertragung findet im Frequenzbereich von $7-14$ Hz statt, das Maximum liegt bei ca. 12.5 Hz.
Auch in diesen Rechnungen stimmen die Berechnungen gut mit den Werten von Freudenberg überein.

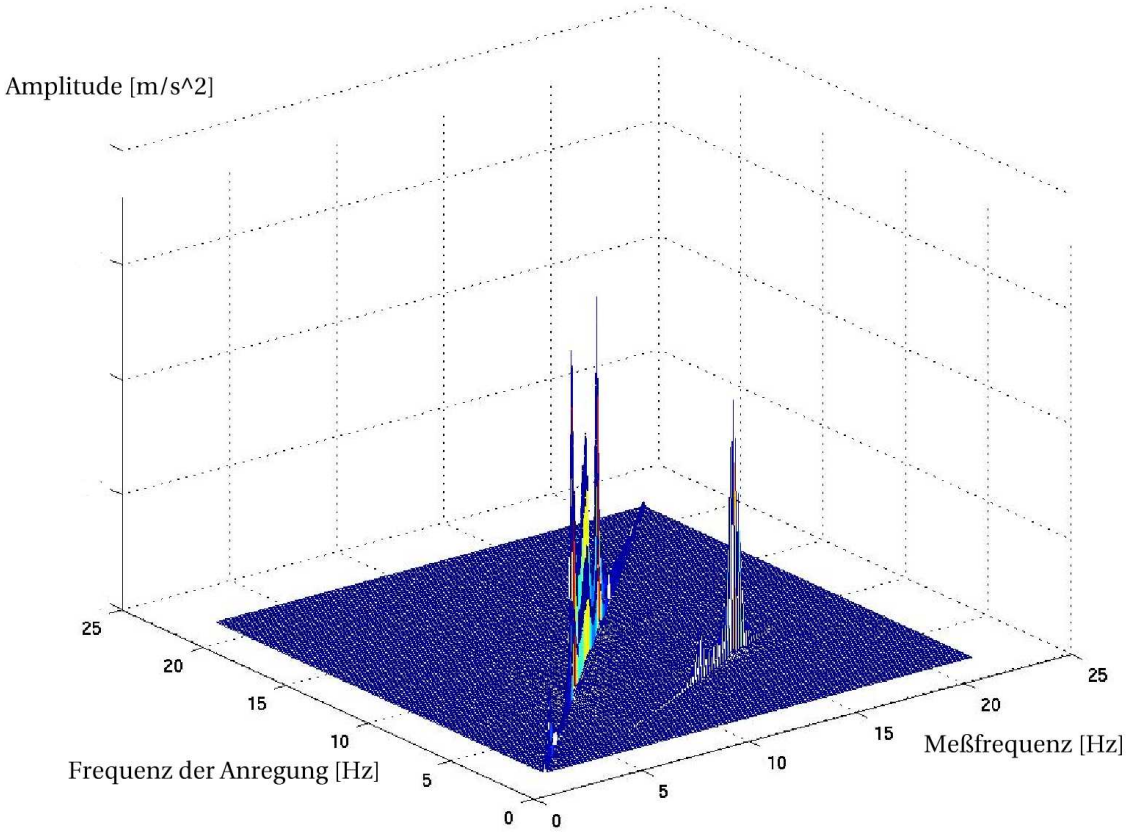

Abbildung 4.6: Fourier-Analyse der Beschleunigungen am Fahrersitz bei Motoranregung

Man sieht die Grundschwingung (Hauptdiagonale) und die erste Oberschwingung (Nebendiagonale)

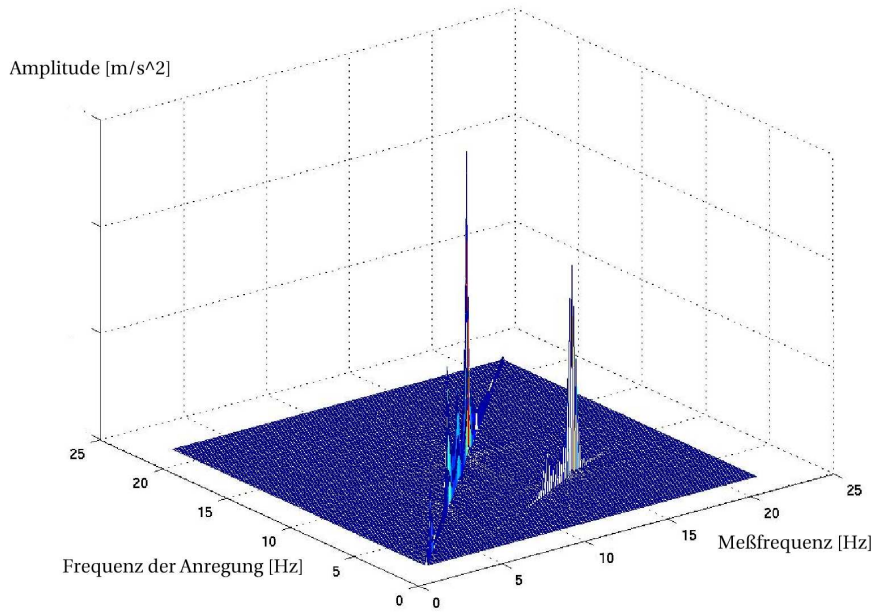

Abbildung 4.7: Fourier-Analyse der Beschleunigungen am Fahrersitz in x-Richtung

4.3.2 Formulierung des Zielfunktionals

Anhand der Erkenntnisse der Simulationen und der Analyse der übertragenen Bewegung am Fahrersitz formulieren wir nun eine Zielfunktion für die Optimierung. Dazu filtern wir aus der Bewegung am Fahrersitz mithilfe einer Fouriertranformation die Grundschwingung (s_α, c_α) und die erste Oberschwingung (s'_α, c'_α) in die drei Raumrichtungen ($\alpha \in \{x,y,z\}$) heraus:

$$
\begin{aligned}
s_\alpha &:= \int_0^{N*2\pi\nu} \sin[2\pi\nu t] \cdot p_\alpha(t)dt \\
s'_\alpha &:= \int_0^{N*2\pi\nu} \sin[2\pi 2\nu t] \cdot p_\alpha(t)dt \\
c_\alpha &:= \int_0^{N*2\pi\nu} \cos[2\pi\nu t] \cdot p_\alpha(t)dt \\
c'_\alpha &:= \int_0^{N*2\pi\nu} \cos[2\pi 2\nu t] \cdot p_\alpha(t)dt \\
&\alpha \in \{x,y,z\}
\end{aligned}
$$

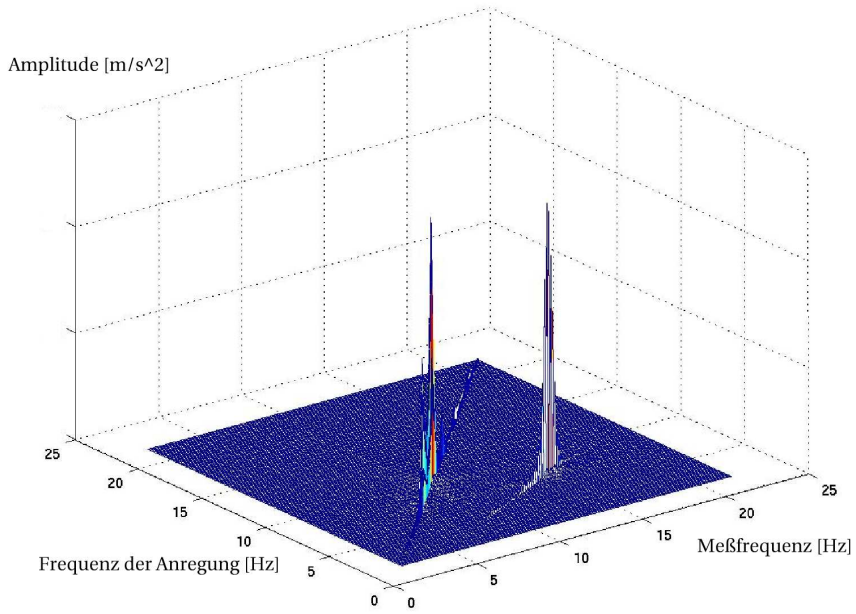

Abbildung 4.8: Fourier-Analyse der Beschleunigungen am Fahrersitz in y-Richtung

Dann multiplizieren wir jeden Term mit einem Gewichtungsfaktor, der von der Frequenz abhängt. Dies ist sinnvoll, da der menschliche Körper Schwingungen verschiedener Frequenz als unterschiedlich unangenehm empfindet. Bestimmte tiefere Frequenzen werden als besonders unangenehm empfunden, daher auch die Tendenz, auf schwankenden Schiffen seekrank zu werden. Richtlinie hierbei ist die Norm VDI 2057 (Einwirkungen mechanischer Schwingungen auf den Menschen - Ganzkörperschwingungen, [VG05]). Abbildung 4.15 zeigt die Frequenzbewertung in z-Richtung für den Bereich $1 - 400$ Hz.
Die Zielfunktion eines Setpoints i mit Anregefrequenz ν schreibt sich nun als:

$$\Phi_i \quad = \quad \sum_{\alpha=x,y,z} (\omega_\nu s_\alpha)^2 + (\omega_{2\nu} s'_\alpha)^2 + (\omega_\nu c_\alpha)^2 + (\omega_{2\nu} c'_\alpha)^2$$

Ziel der Optimierung ist eine Verbesserung des Fahrkomforts. Die Schwingungsübertragung zum Fahrersitz soll also gering sein für alle Arten von Anregungen, das heißt für alle Rad- und Motoranregungen im vorgegebenen Frequenzbereich (2-20Hz). Um dies zu erreichen, ist eine simultane Optimierung von mehreren *Setpoints* nötig. Für die Multiple-Setpoint-Optimierung summieren wir die Terme für alle Anregungen auf:

$$\Phi \quad = \quad \sum_{i=1}^{N} \Phi_i$$

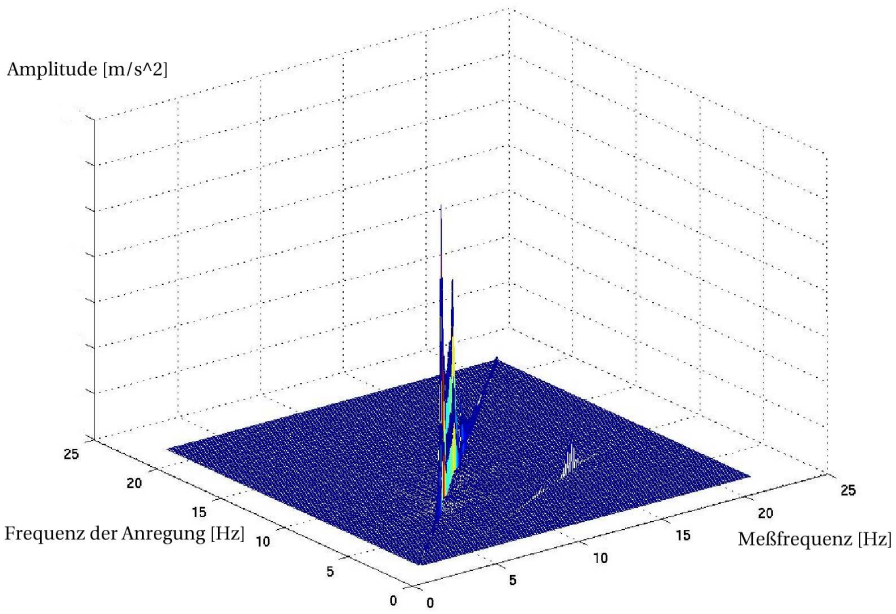

Abbildung 4.9: Fourier-Analyse der Beschleunigungen am Fahrersitz in z-Richtung

4.3.3 Ergebnisse der Optimierung

Optimierungsparameter sind die Positionen bzw. die Steifigkeiten der Lager. Es ergibt sich folgende mathematische Problemformulierung:

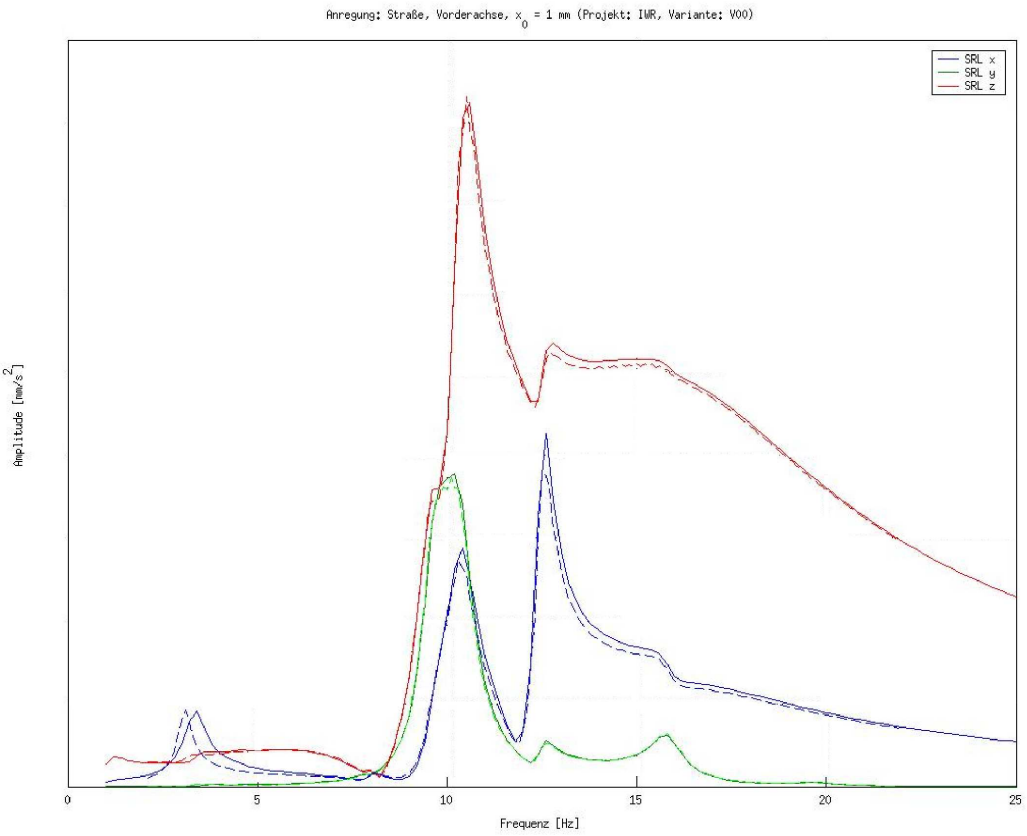

Abbildung 4.10: Ergebnisse der Simulation: Beschleunigungen am Fahrersitz bei Anregung der Vorderachse mit 1mm Amplitude
gestrichelte Kurve: Berechnungen von Freudenberg
durchgezogene Kurve: eigene Ergebnisse

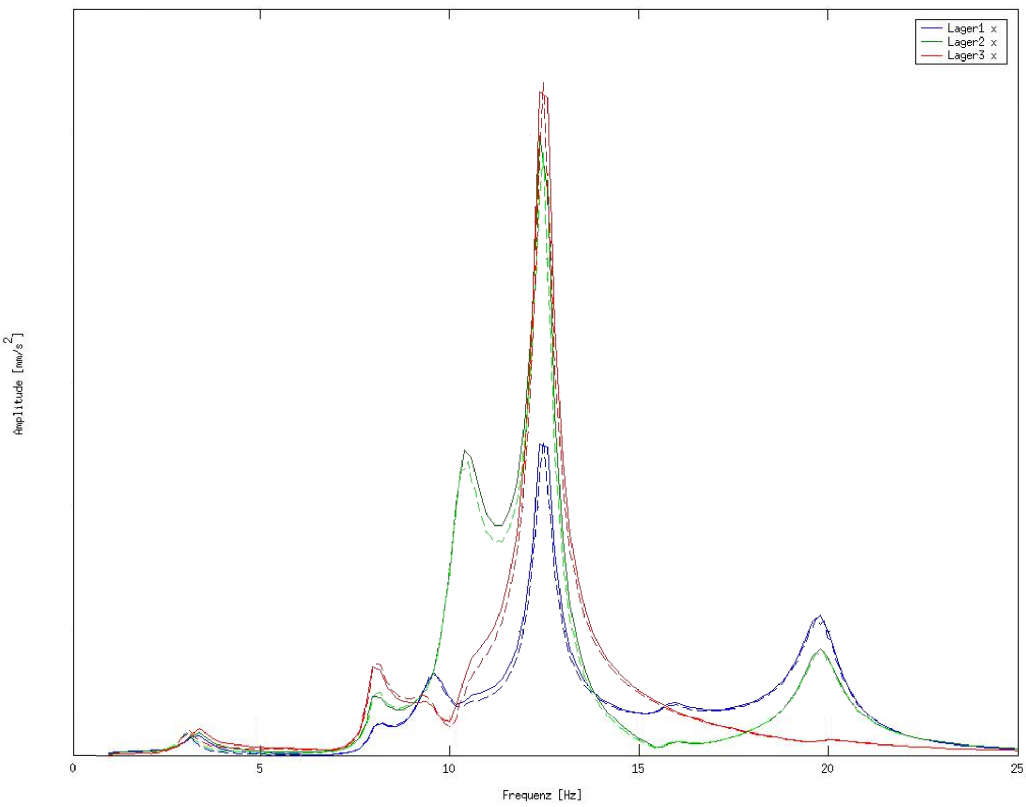

Abbildung 4.11: Ergebnisse der Simulation: Beschleunigungen in den Motorlagern in x-Richtung bei Anregung der Vorderachse mit 1mm Amplitude
gestrichelte Kurve: Berechnungen von Freudenberg
durchgezogene Kurve: eigene Ergebnisse

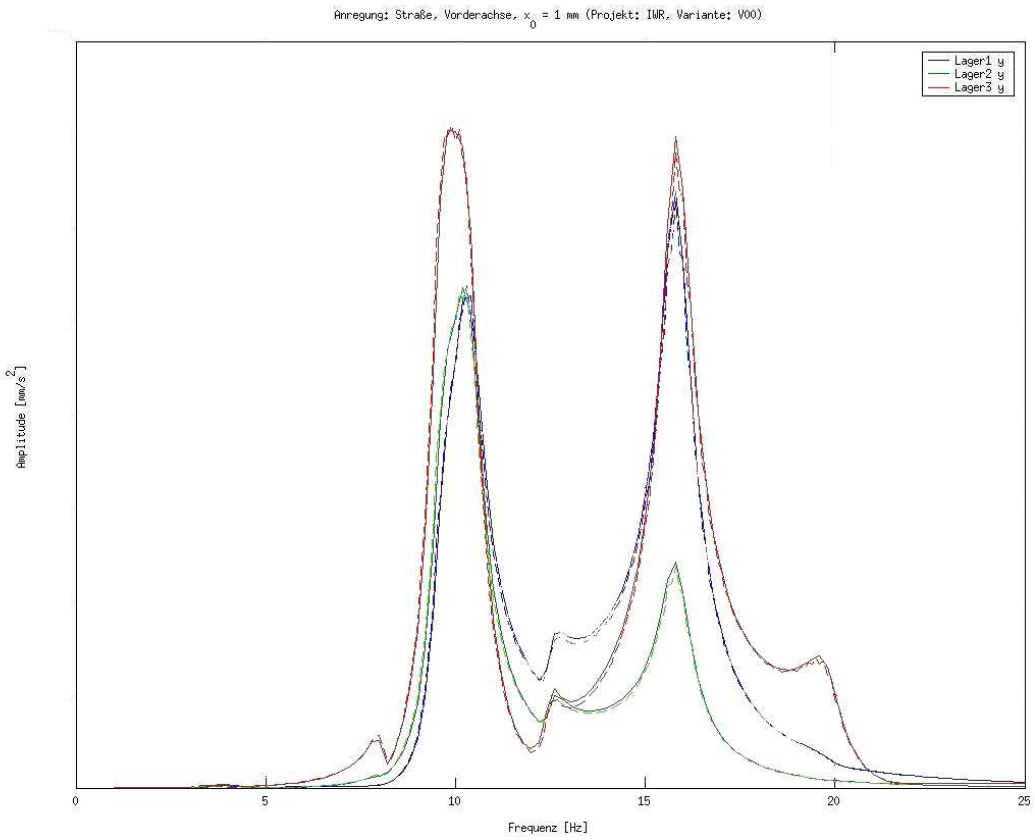

Abbildung 4.12: Ergebnisse der Simulation: Beschleunigungen in den Motorlagern in y-Richtung bei Anregung der Vorderachse mit 1mm Amplitude
gestrichelte Kurve: Berechnungen von Freudenberg
durchgezogene Kurve: eigene Ergebnisse

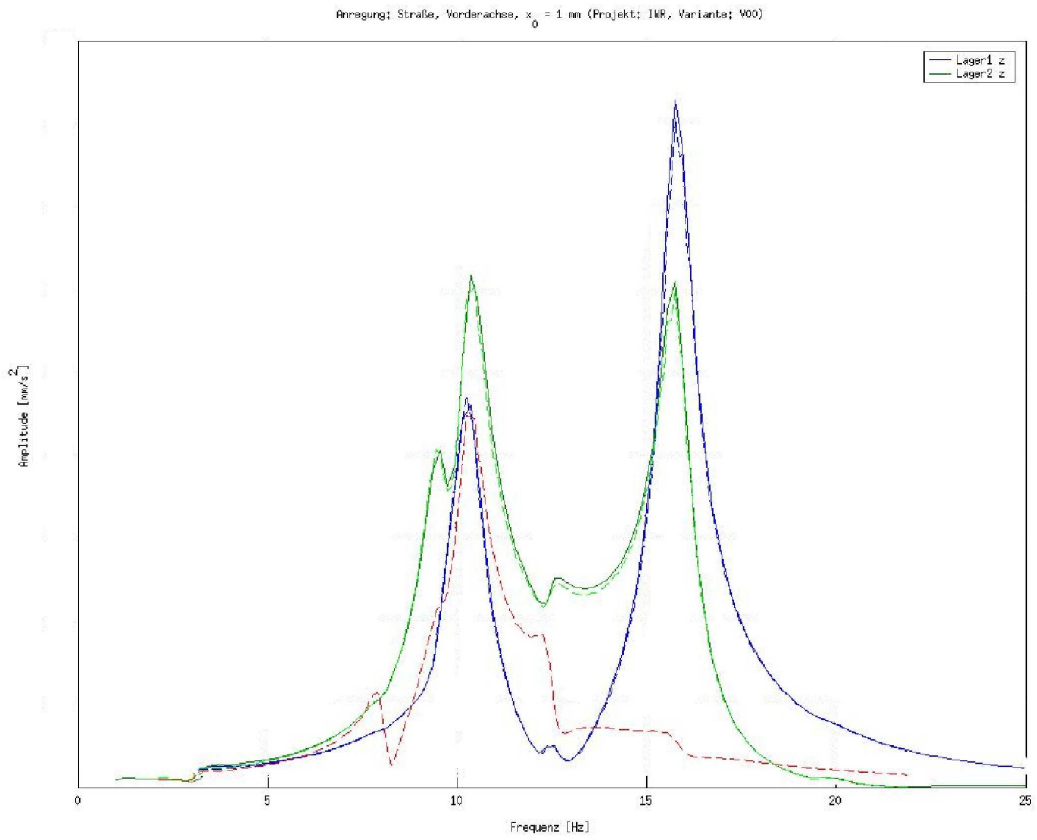

Abbildung 4.13: Ergebnisse der Simulation: Beschleunigungen in den Motorlagern in z-Richtung bei Anregung der Vorderachse mit 1mm Amplitude
gestrichelte Kurve: Berechnungen von Freudenberg
durchgezogene Kurve: eigene Ergebnisse

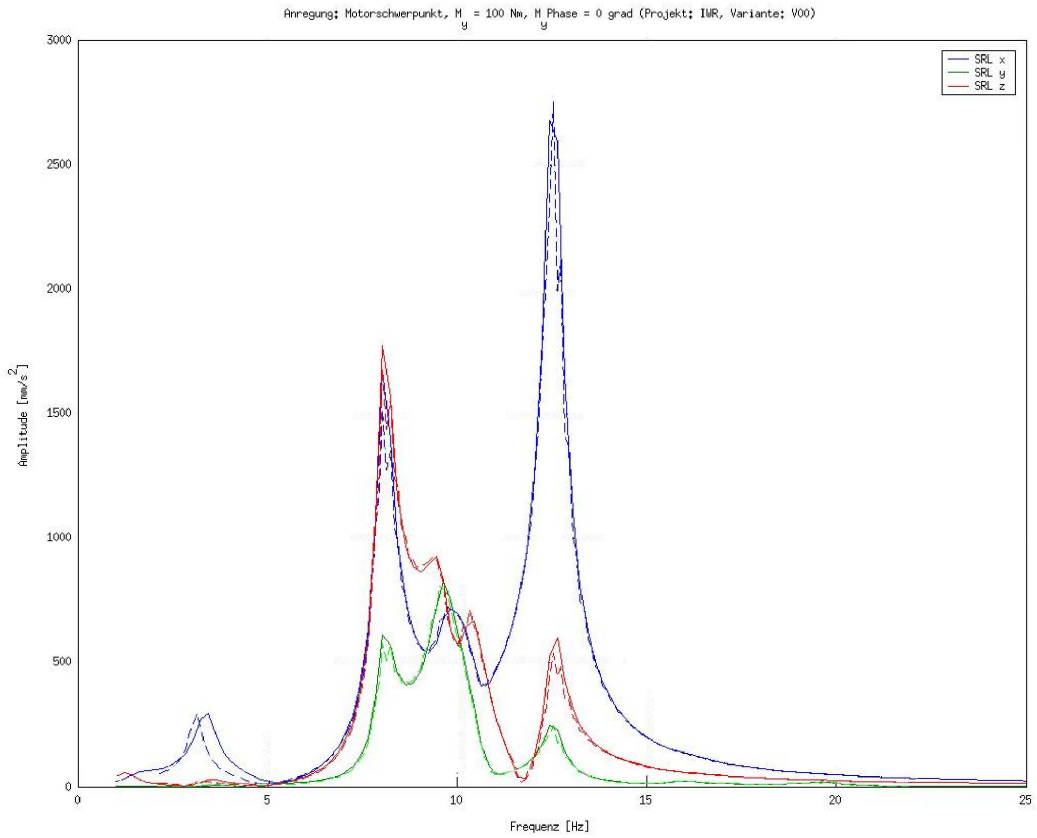

Abbildung 4.14: Ergebnisse der Simulation: Beschleunigungen am Fahrersitz bei Motoranregung
gestrichelte Kurve: Berechnungen von Freudenberg
durchgezogene Kurve: eigene Ergebnisse

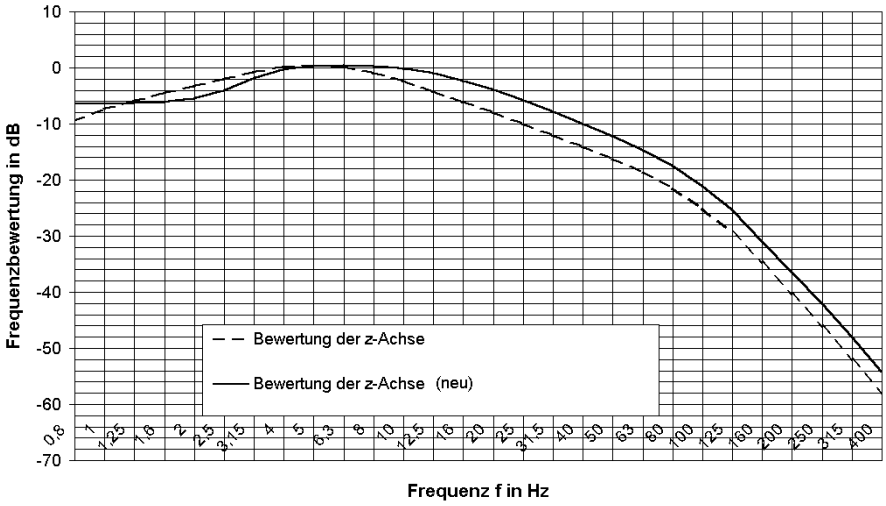

Abbildung 4.15: Frequenzbewertung nach VDI 2057

$$\min_{p} \sum_{\nu} \Phi_{\nu}$$

unter den folgenden Nebenbedingungen:

DAE-Modellgleichungen

$$
\begin{aligned}
\dot{e}_i &= \breve{f}_i(t,y_i,v_i,e_i) \\
\dot{y}_i &= v_i \\
\dot{v}_i &= a_i \\
\begin{pmatrix} M_i(y_i,e_i) & G_i(y_i,e_i)^T \\ G_i(y_i,e_i) & 0 \end{pmatrix} \begin{pmatrix} a_i \\ \lambda_i \end{pmatrix} &= \begin{pmatrix} f_i(t,y_i,v_i,e_i) \\ \gamma_i(t,y_i,v_i,e_i) \end{pmatrix} \\
g_i(y_i) &= 0 \\
G_i(y_i)v_i &= 0
\end{aligned}
$$

für $t \in \left[0,\frac{1}{\nu}\right]$
feste Anfangswerte

$$
\begin{aligned}
y_i(0) &= y_{i,0} \\
v_i(0) &= v_{i,0}
\end{aligned}
$$

Schranken an Motorverdrehung und Parameter

$$
\begin{aligned}
-h_i(p_i(t)) + \beta_i &\geq 0 \\
-h_i(p_i(t)) - \beta_i &\geq 0 \\
p_i - \underline{p} &\geq 0 \\
-p_i + \bar{p} &\geq 0
\end{aligned}
$$

} für $i = 0, \ldots, N-1$

Beginnen wir mit der einfachsten Rechnung, einer Single-Setpoint-Optimierung für eine bestimmte Anregung mit festgelegter Frequenz.
Bei der Optimierung der Positionen ist die Größe des entstehenden Optimierungsproblems wie folgt:

- 150 differentielle Variablen (32 Freiheitsgrade, 112 Invarianten, 6 externe Variablen)
- 54 gekoppelte Parameter mit 9 Freiheitsgraden
- 45 Nebenbedingungen für die Kopplung der Parameter
- 3 weitere Nebenbedingungen an die Motorverdrehung

Bei der Optimierung der Steifigkeiten haben wir:

- 150 differentielle Variablen (32 Freiheitsgrade, 112 Invarianten, 6 externe Variablen)
- 7 entkoppelte Parameter

- 3 Nebenbedingungen an Motorverdrehung

In beiden Fällen ergibt sich bei einer Diskretisierung in zehn Multiple-Shooting-Intervallen eine Dimension des Optimierungsproblems von 1614 Variablen.

Bei der Optimierung der Positionen ist die Verschiebung der Lager begrenzt auf 3 cm in jede Richtung ausgehend von der Position des Startmodells von Freudenberg.
Bei der Optimierung der Steifigkeiten ist die Federkonstante auf den Bereich $50 - 500$ N/mm beschränkt. Als Dauer für Einschwingphase und Meßphaße verwenden wir jeweils 1 Sekunde.
Die Abbildungen 4.16 und 4.17 zeigen den Vergleich der Trajektorien des Fahrersitzes vor und nach einer Optimierung bei einer Radanregung von 10.5 Hz.

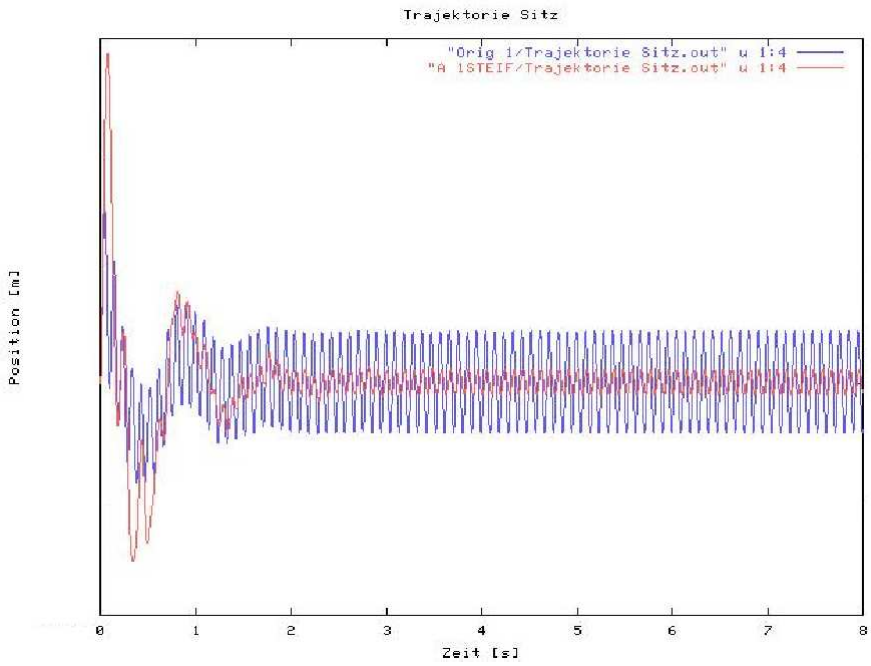

Abbildung 4.16: Ergebnisse der Optimierung: Vergleich der Trajektorien nach Optimierung der Steifigkeiten bei Radanregung von 10.5 Hz

Die Abbildungen 4.18 und 4.19 zeigen den Vergleich der Trajektorien des Fahrersitzes vor und nach einer Optimierung bei einer Motoranregung von 10.5 Hz.

Die Rechenzeiten für Optimierungen für eine Single-Setpoint-Rechnung betrugen 30-120 Minuten, je nach Anzahl der SQP-Iterationen (10-50 Iterationen), je nach Wahl der Startwerte, der Anregung und der Skalierungsfaktoren.
Führt man für die optimierten Lösungen einen kompletten Sweep (Kette von Simulationen im ganzen Frequenzbereich) durch, sieht man, daß die Schwingungsübertragung für die gewählte Anregung deutlich minimiert wird, aber auch eine globale Verbesserung der Zielfunktion im

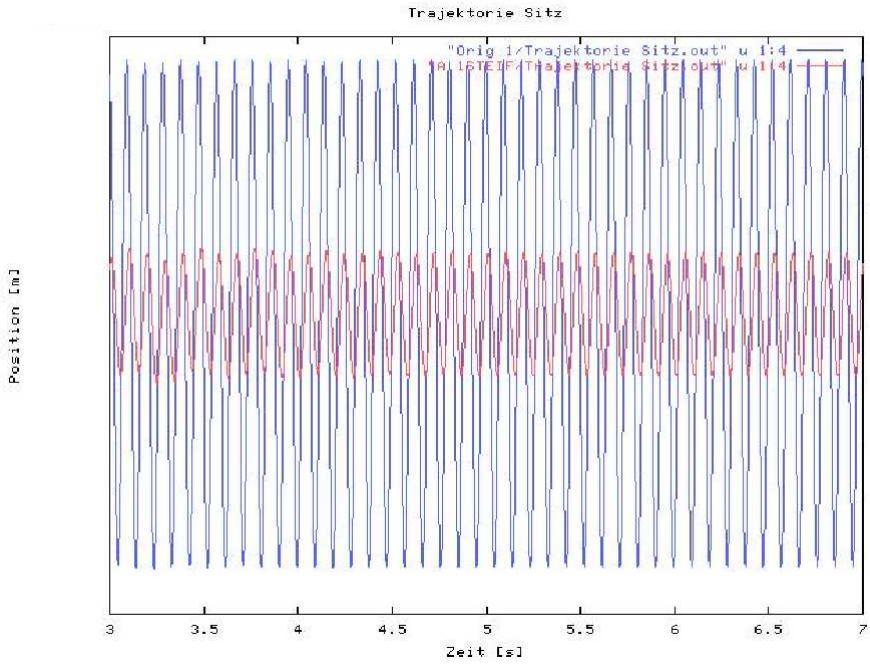

Abbildung 4.17: Ergebnisse der Optimierung: Vergleich der Trajektorien nach Optimierung der Steifigkeiten bei Radanregung von 10.5 Hz (Detailansicht)

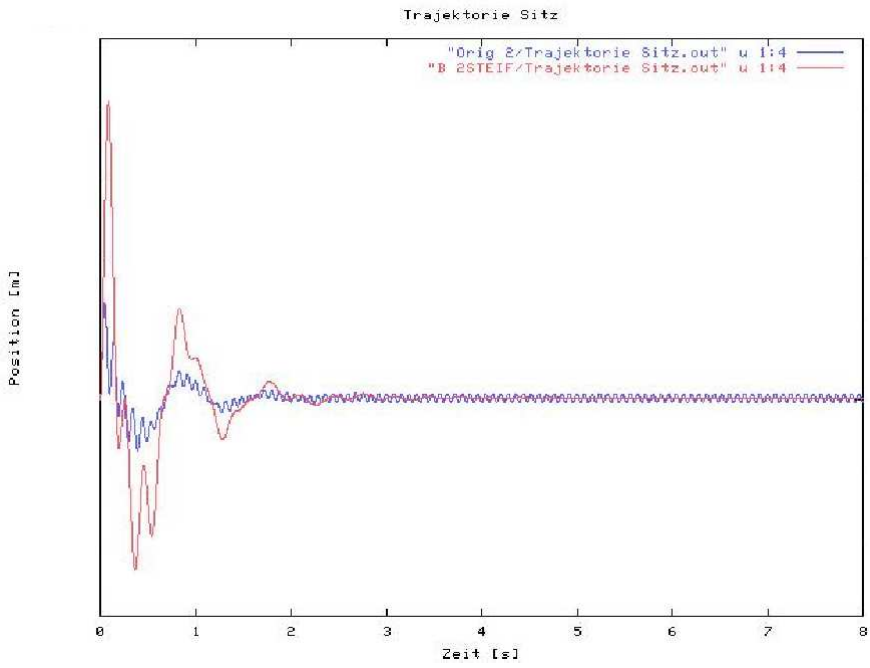

Abbildung 4.18: Ergebnisse der Optimierung: Vergleich der Trajektorien nach Optimierung der Steifigkeiten bei Motoranregung von 12.5 Hz

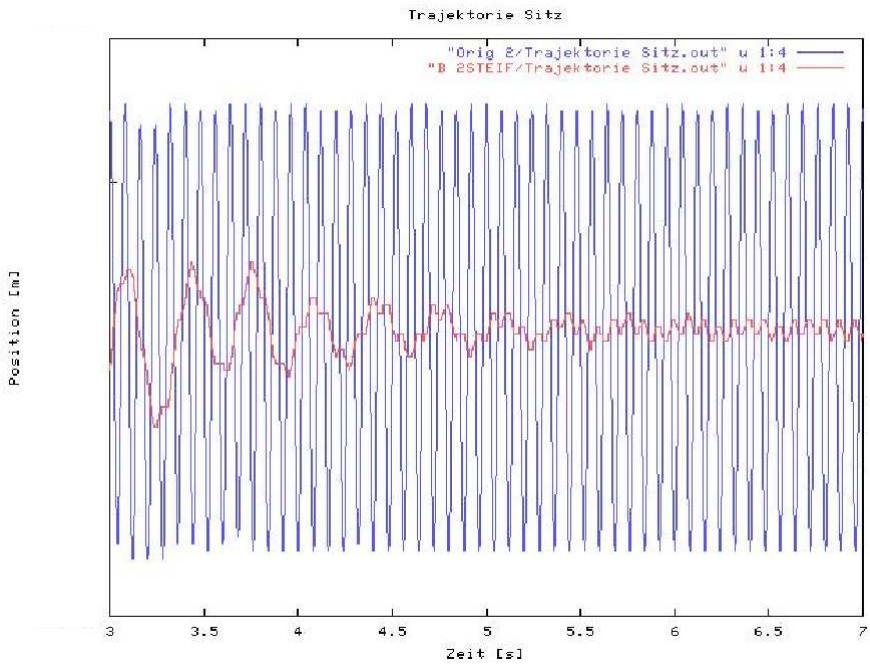

Abbildung 4.19: Ergebnisse der Optimierung: Vergleich der Trajektorien nach Optimierung der Steifigkeiten bei Motoranregung von 12.5 Hz (Detailansicht)

gesamten Frequenzbereich erreicht wird. Die Multiple-Setpoint-Optimierung wird uns zeigen, daß die Schwingungsübertragung im gesamten Frequenzbereich noch weiter verbessert werden kann.
Abbildung 4.20 und 4.21 zeigen das Verhalten der für die Radanregung beziehungsweise für die Motoranregung optimierten Lösungen, wobei der Pfeil die für die Optimierung gewählte Frequenz anzeigt.

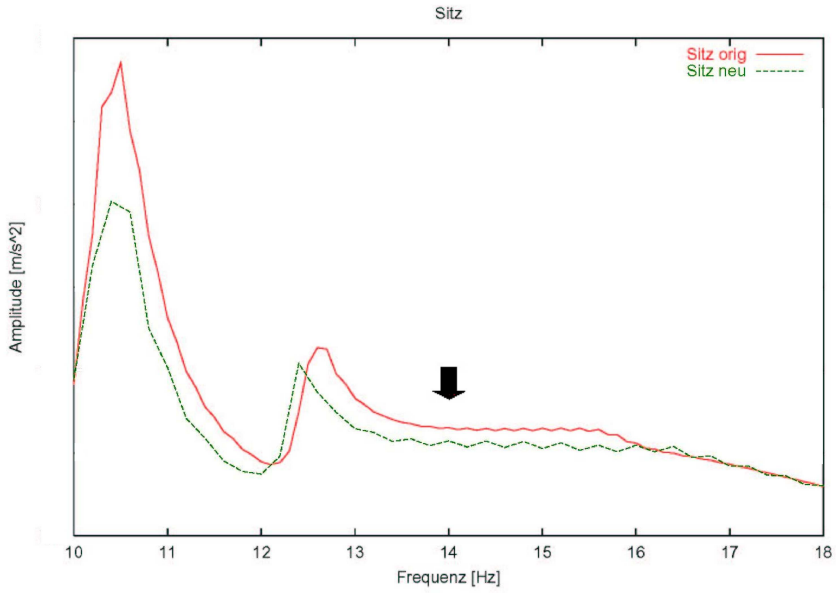

Abbildung 4.20: Optimierte Aufhängung für Radanregung

Die Lösung der Single-Setpoint-Optimierung nehmen wir jetzt als Startlösung für eine Multiple-Setpoint-Optimierung. Wir betrachten nun eine simultane Optimierung einer ganzen Schar von Anregungen:

1. nur Gravitation
2. Radanregung mit variabler Frequenz
3. Motoranregung mit variabler Frequenz
4. konstantes Motordrehmoment

Neben der Optimierung des Fahrkomforts bei 2. und 3. verlangen wir eine Kontrolle über die maximale Motorverdrehung in allen vier Fällen.

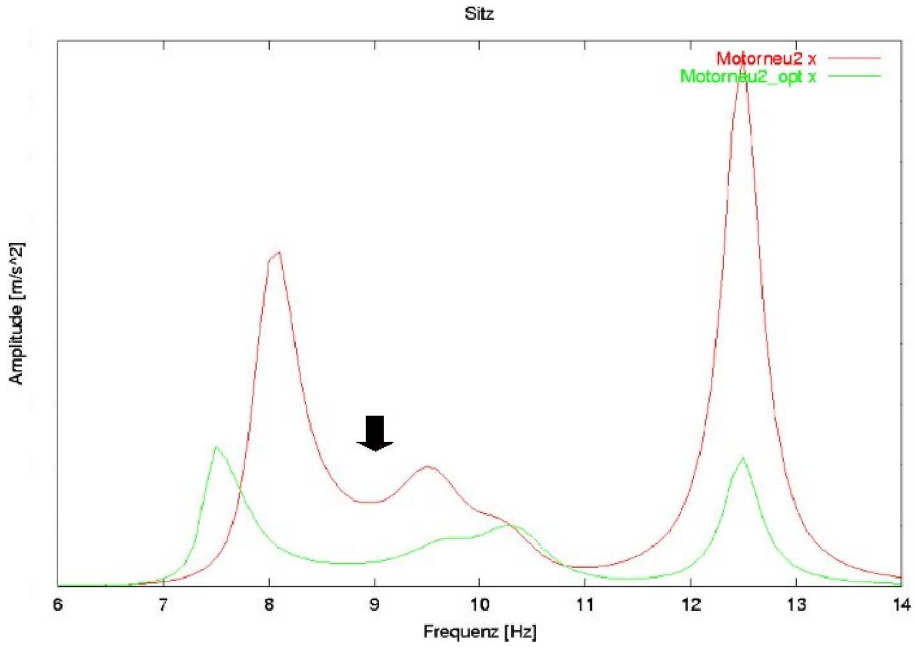

Abbildung 4.21: Optimierte Aufhängung für Motoranregung

Abbildung 4.22 zeigt das Ergebnis einer Multiple-Setpoint-Optimierung im Vergleich zu den Lösungen der einzelnen Setpoints. Optimiert wurden die Positionen der Motorlager für eine Motoranregung von $8{,}9{,}10$ und 11 Hz. Die Multiple-Setpoint-Optimierung brauchte 22 Iterationen und eine Rechenzeit von ca. 4 Stunden. Das Ergebnis ist eine modifizierte Aufhängung, die die Vibrationsübertragung im Frequenzbereich $8 - 11$ Hz weiter verringert.

Abbildung 4.23 sowie folgende Tabelle zeigen die Zielfunktionswerte der Lösungen im Vergleich. Jede Single-Setpoint-Lösung liefert für ihre Anregungsfrequenz den besten Zielfunktionswert. Für andere Frequenzen verhält sie sich jedoch unter Umständen nicht wie gewünscht. Die Lösung der Multiple-Setpoint-Optimierung liefert für alle vier Anregungen niedrige Zielfunktionswerte.

Zielfunktion bei	8Hz	9Hz	10Hz	11 Hz	Quadratsumme
Startlösung	2.309	1.173	1.109	0.448	8.25
8Hz-Lösung	**0.225**	0.267	0.306	0.256	0.28
9Hz-Lösung	0.294	**0.202**	0.300	0.219	0.27
10Hz-Lösung	0.497	0.486	**0.215**	0.187	0.56
11Hz-Lösung	0.916	0.602	0.239	**0.113**	1.27
MS-Lösung	0.260	0.251	0.258	0.164	**0.22**

Die nächste Rechnung verbindet alle vier Arten von Anregung. Wir haben einen Setpoint mit reiner Gravitation, drei Setpoints mit Radanregung (9 Hz,10 Hz,11 Hz), drei Setpoints mit Motoranregung (9 Hz, 10 Hz, 11 Hz) und einen Setpoint mit konstantem Motormoment (600 Nm).

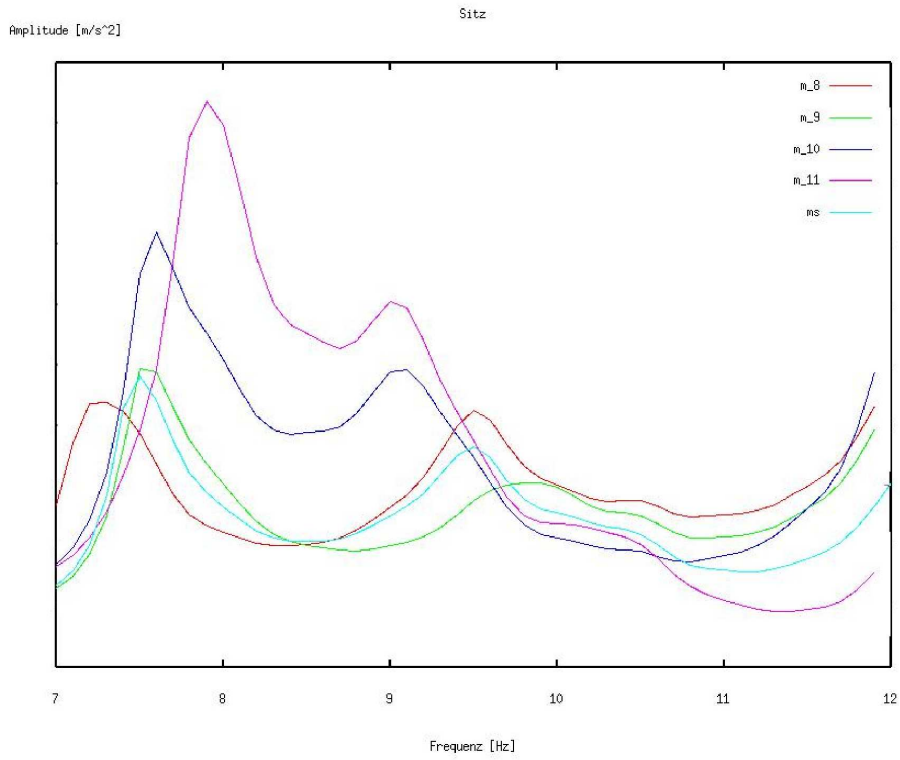

Abbildung 4.22: Vergleich: Multiple-Setpoint-Optimierung – Single-Setpoint-Optimierung

Die türkise Multiple-Setpoint-Lösung bietet die beste Dämpfung im gegebenen Frequenzbereich

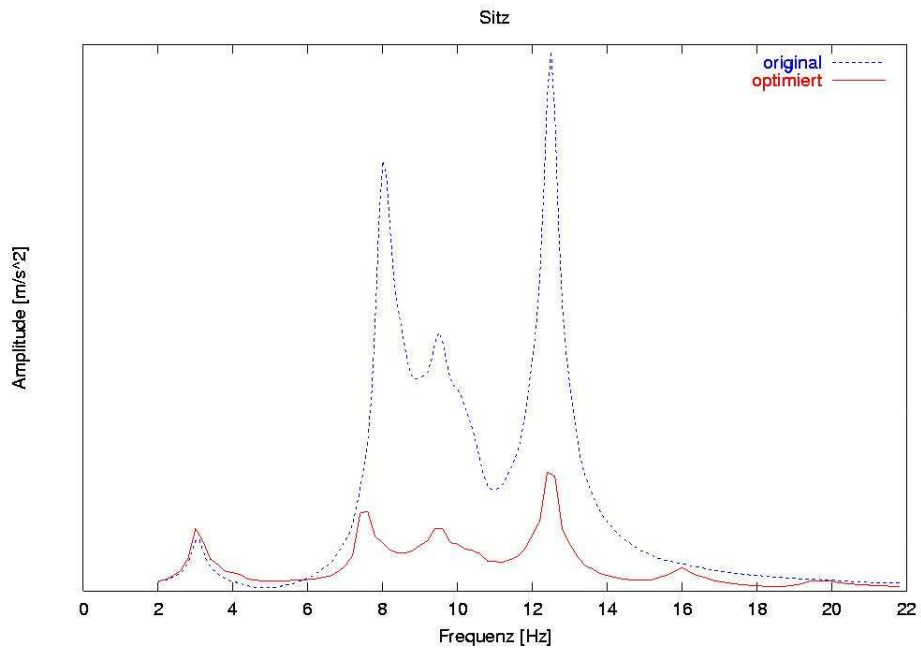

Abbildung 4.23: Multiple-Setpoint-Lösung 1 bei Motoranregung im Vergleich zur Startlösung

Nach 10.5 Stunden Rechenzeit und 29 SQP-Iterationen wird die Lösung berechnet, die in Abbildung 4.24 und 4.25 dargestellt ist.

Eine ähnliche Rechnung lieferte nach 9.5 Stunden Rechenzeit und 32 SQP-Iterationen die Lösung für folgende Setpoints: Einen Setpoint mit reiner Gravitation, drei Setpoints mit Radanregung (10 Hz, 12 Hz, 14 Hz), drei Setpoints mit Motoranregung (9 Hz, 10 Hz, 11 Hz) und einen Setpoint mit konstantem Motormoment. Die Lösung ist in Abbildung 4.26 und 4.27 dargestellt.

4.3.4 Ergebnisse der Optimierung – Randwertproblemansatz

Der bisherige Optimierungansatz über die Formulierung eines Anfangswertproblems ist zwar der übliche, er bringt jedoch einige Schwierigkeiten mit sich, die mit dem Einschwingvorgang (siehe Abschnitt 4.3.1) zu tun haben. Die Fourier-Analyse, die für die Formulierung der Zielfunktion nötig ist, setzt einen komplett eingeschwungenen Zustand voraus. Doch selbst wenn man sehr lange Zeit zum Einschwingen einräumt, bleibt die anschließend betrachtete Schwingung eine Überlagerung von Einschwingvorgang und angeregter Schwingung. Andererseits ist es für eine Optimierung in akzeptabler Rechenzeit wichtig, die Integrationszeiten möglichst kurz zu halten. In jedem Fall kann der stationäre Zustand in der Einschwingphase nur approximativ erreicht werden. Ein weiteres Problem ist, daß sich der stationär eingeschwungene Zustand im Laufe der Optimierung durch das Verschieben der Lager ändert, sodaß der Einfluß des Ein-

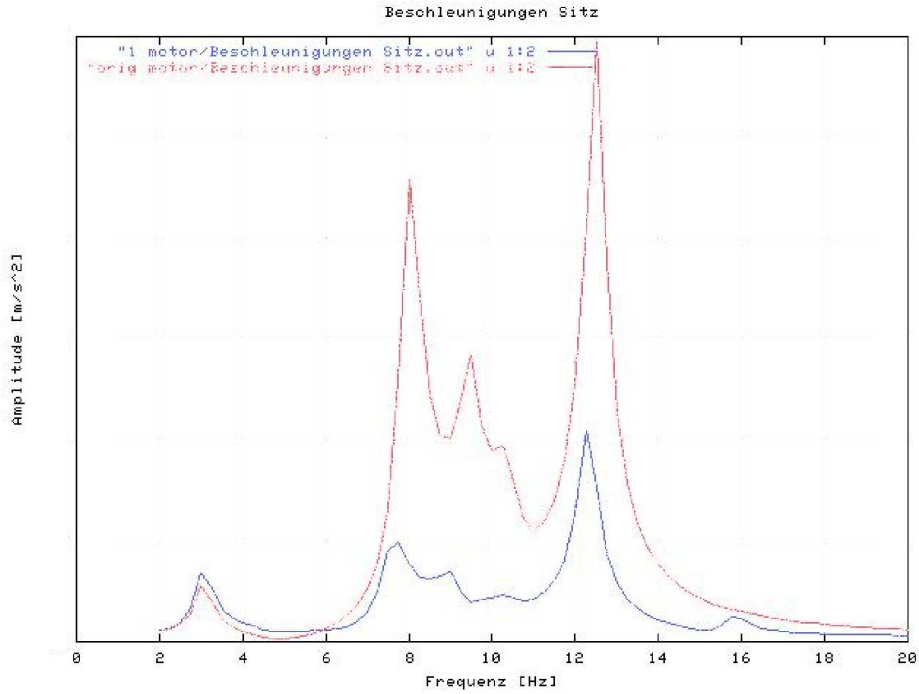

Abbildung 4.24: Multiple-Setpoint-Lösung 2 im Vergleich zur Startlösung bei Motoranregung

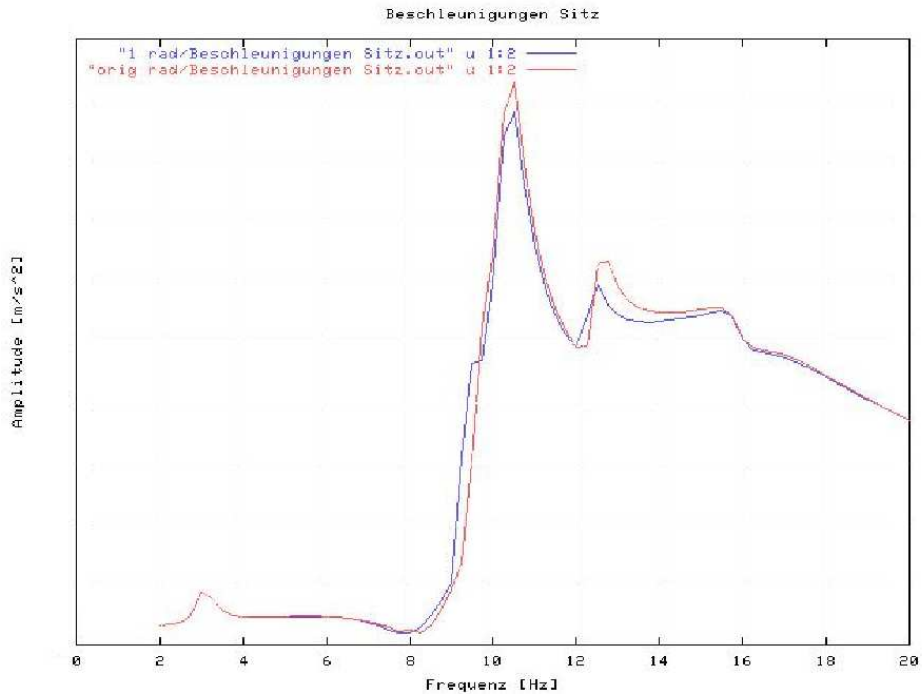

Abbildung 4.25: Multiple-Setpoint-Lösung 2 im Vergleich zur Startlösung bei Radanregung

schwingvorgangs zunimmt, da die festen Anfangswerte immer weiter entfernt sind vom stationär eingeschwungenen Zustand.
Da das Problem an sich schon einige Schwierigkeiten aufbietet,

- unterschiedliche Größenverhältnisse von Modell (circa 3 Meter Länge) und Anregung (1 mm) beziehungsweise übertragene Schwingung (< 1 mm)
- hohe Integrationsgenauigkeit nötig zum Nachfahren zahlreicher Perioden von kleinen Schwingungen
- nicht Standard-Zielfunktion (Fourier-Analyse über Trajektorie)
- große Zahl an Optimierungsvariablen durch die Multiple-Setpoint-Struktur

ergeben sich für eine größere Anzahl Setpoint schnell immens hohe Rechenzeiten (z.B. 17 Tage Rechenzeit für 80 Setpoints), außerdem treten mitunter numerische Schwierigkeiten auf (keine Konvergenz).
Abhilfe schafft ein neuer Ansatz, der auf der Lösung von Randwertproblemen beruht. Die Integrationsdauern können so deutlich reduziert werden. Statt festen Anfangswerten verwenden wir freie Anfangswerte, das heißt die Anfangswerte sind zusätzliche Optimierungsparameter. Integriert wird nur noch über eine Periode der anregenden Schwingung. Der eingeschwungene Zustand wird über zusätzliche Randbedingungen erreicht, die eine Periodizität der Bewegung

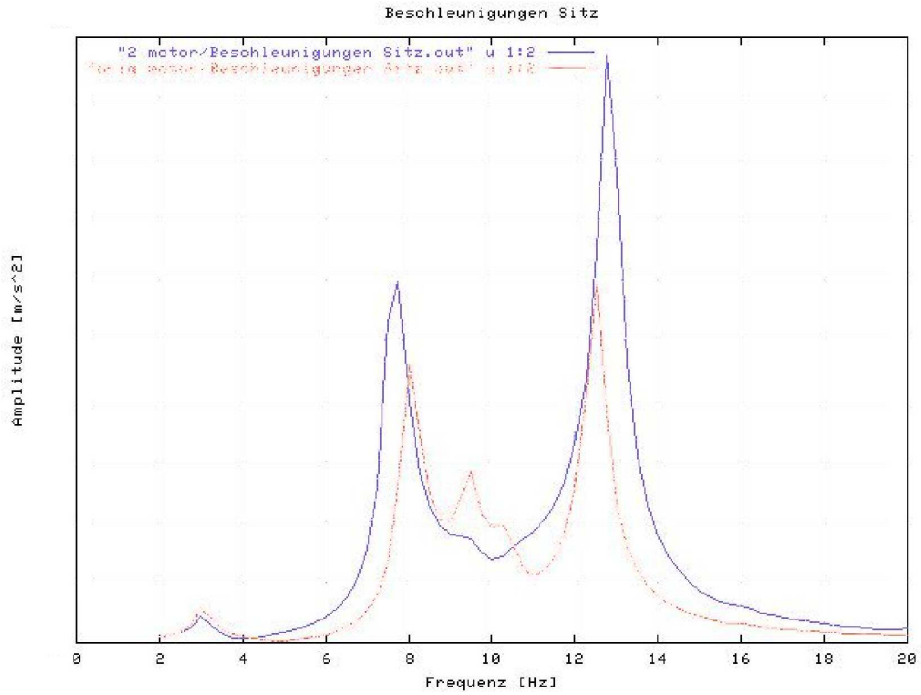

Abbildung 4.26: Multiple-Setpoint-Lösung 3 im Vergleich zur Startlösung bei Motoranregung

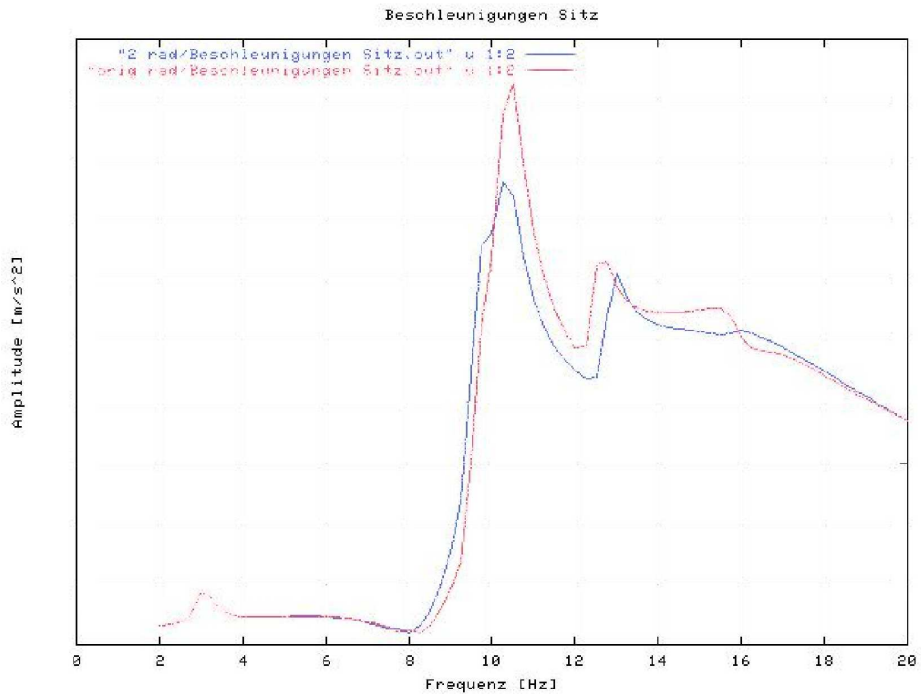

Abbildung 4.27: Multiple-Setpoint-Lösung 3 im Vergleich zur Startlösung bei Radanregung

garantieren, das heißt Endpositionen und Endgeschwindigkeiten müssen identisch sein mit den Anfangspositionen und Anfangsgeschwindigkeiten.

Es ergibt sich folgende Problemformulierung:

$$\min_{y_i(0),v_i(0),p} \sum_{\nu} \Phi_\nu$$

unter den folgenden Nebenbedingungen:

DAE-Modellgleichungen

$$\begin{aligned}
\dot{e}_i &= \breve{f}_i(t,y_i,v_i,e_i) \\
\dot{y}_i &= v_i \\
\dot{v}_i &= a_i \\
\begin{pmatrix} M_i(y_i,e_i) & G_i(y_i,e_i)^T \\ G_i(y_i,e_i) & 0 \end{pmatrix} \begin{pmatrix} a_i \\ \lambda_i \end{pmatrix} &= \begin{pmatrix} f_i(t,y_i,v_i,e_i) \\ \gamma_i(t,y_i,v_i,e_i) \end{pmatrix} \\
g_i(y_i) &= 0 \\
G_i(y_i)v_i &= 0
\end{aligned}$$

für $t \in \left[0,\frac{1}{\nu}\right]$

Schranken an Motorverdrehung und Parameter

$$\begin{aligned}
-h_i(p_i(t)) + \beta_i &\geq 0 \\
-h_i(p_i(t)) - \beta_i &\geq 0 \\
p_i - \underline{p} &\geq 0 \\
-p_i + \bar{p} &\geq 0
\end{aligned}$$

periodische Randbedingungen

$$\begin{aligned}
y_i(0) &= y_i\left(\frac{1}{\nu}\right) \\
v_i(0) &= v_i\left(\frac{1}{\nu}\right)
\end{aligned}$$

} für $i = 0, \ldots, N-1$

Diese Methode erhöht zwar die Dimension des Optimierungsproblem – statt 7 (Steifigkeiten) bzw. 54 (Positionen) Optimierungsparameter haben wir nun $7 + 144 = 153$ bzw. $54 + 144 = 198$ Optimierungsparameter. Außerdem kommen 144 Gleichungsnebenbedingungen für die Periodizität dazu. Da bei der bisherigen Methode über 95% der Zeit für die Integration zu Buche standen, ist die modifizierte Problemformulierung trotz der Vergrößerungen des Problems angesichts der drastischen Reduzierung der Integrationszeiten in Hinblick auf die Rechenzeit von großem Nutzen. Statt mehreren Sekunden Integration genügt nun für jeden Setpoint die Integration einer einzigen Periode, also zwischen 0.05 und 0.5 Sekunden, darüber hinaus haben wir in der Lösung keine Überlagerung von Einschwingvorgängen. Die Rechenzeit für die Optimierung eines einzelnen Setpoints verringert sich dadurch um 95%. Für steigende Setpointzahl nimmt der Einfluß

des vergrößerten Systems zu. Bei ca. 15 Setpoints ist die Rechenzeit für das Lösen des quadratischen Programms innerhalb des SQP-Algorithmus' so angewachsen, daß sie genauso viel Zeit in Anspruch nimmt wie die Integration. Die gesamte Rechenzeit verringert sich im Vergleich zur bisherigen Methode also noch um 90%.

Rechnet man dasselbe Multiple-Setpoint-Problem mit beiden Ansätzen, zeigt sich, daß wir in der Lösung identische Werte für die Positionen bzw. Steifigkeiten der Lager erhalten.

Abbildung 4.28 zeigt das Schwingungverhalten im Vergleich zur ursprünglichen Motoraufhängung nach Optimierung für folgende Setpoints:

Setpointzahl	Anregefrequenz der verschiedenen Setpoints
2 Setpoints	8.0Hz, 12.4Hz
4 Setpoints	8.0Hz, 9.1Hz, 10.3Hz, 12.4Hz
8 Setpoints	7.0Hz, 8.0Hz, 8.4Hz, 8.7Hz, 9.1Hz, 10.3Hz, 11.0Hz, 12.4Hz
16 Setpoints	7.0Hz, 7.5Hz, 8.0Hz, 8.2Hz, 8.4Hz, 8.7Hz, 9.1Hz, 10.0Hz, 10.3Hz, 10.6Hz, 11.0Hz, 11.6Hz, 12.2Hz, 12.4Hz, 12.6Hz
32 Setpoints	7.0 Hz, 7.5 Hz, 7.8 Hz, 8.0 Hz, 8.2 Hz, 8.4 Hz, 8.5 Hz, 8.6 Hz, 8.7 Hz, 8.8 Hz, 8.9 Hz, 9.0 Hz, 9.1 Hz, 9.2 Hz, 9.4 Hz, 9.6 Hz, 9.8 Hz, 10.0 Hz, 10.1 Hz, 10.2 Hz, 10.3 Hz, 10.4 Hz, 10.5 Hz, 10.6 Hz, 10.8 Hz, 11.0 Hz, 11.3 Hz, 11.6 Hz, 12.2 Hz, 12.4 Hz, 12.6 Hz, 12.8 Hz

Durch die Optimierungen konnte eine Motorlagerung gefunden werden, die bei Motoranregung die Schwingungsübertragung im vorgegebenen Frequenzbereich um ca. 80% (!) reduziert.

Bei den Optimierungen ist eine geschickte Wahl der Setpoints nützlich. Abbildung 4.29 zeigt die Ergebnisse zweier Rechnungen mit jeweils acht Setpoints, einmal mit äquidistanten Anregefrequenzen und einmal mit Anregefrequenzen, die dort konzentriert wurden, wo große Schwingungsübertragung zu erwarten war.

Die Optimierungen benötigen im Regelfall 10-30 SQP-Iterationen. Die Rechenzeiten liegen deutlich niedriger als beim herkömmlichen Ansatz, wie die folgende Tabelle zeigt. Betrachtet wurden dreizehn verschiedene Optimierungen mit einer Setpointzahl von $2-32$. Während beim bisherigen Ansatz mit einer Rechenzeit für die Optimierung von ca. einer Stunde pro Setpoint gerechnet werden mußte, kann jetzt in einer Stunde ein Optimierungsproblem mit $15-16$ Setpoints gelöst werden.

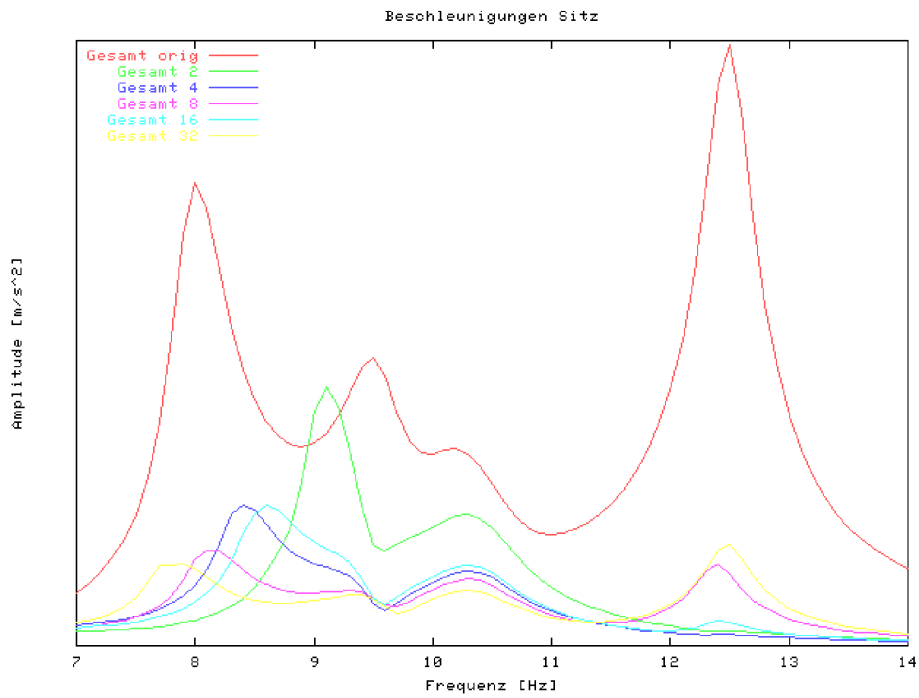

Abbildung 4.28: Multiple-Setpoint-Lösungen bei Randwertproblem-Ansatz

Die Schwingungsübertragung konnte im vorgegebenen Frequenzbereich um ca. 80% (!) reduziert werden.

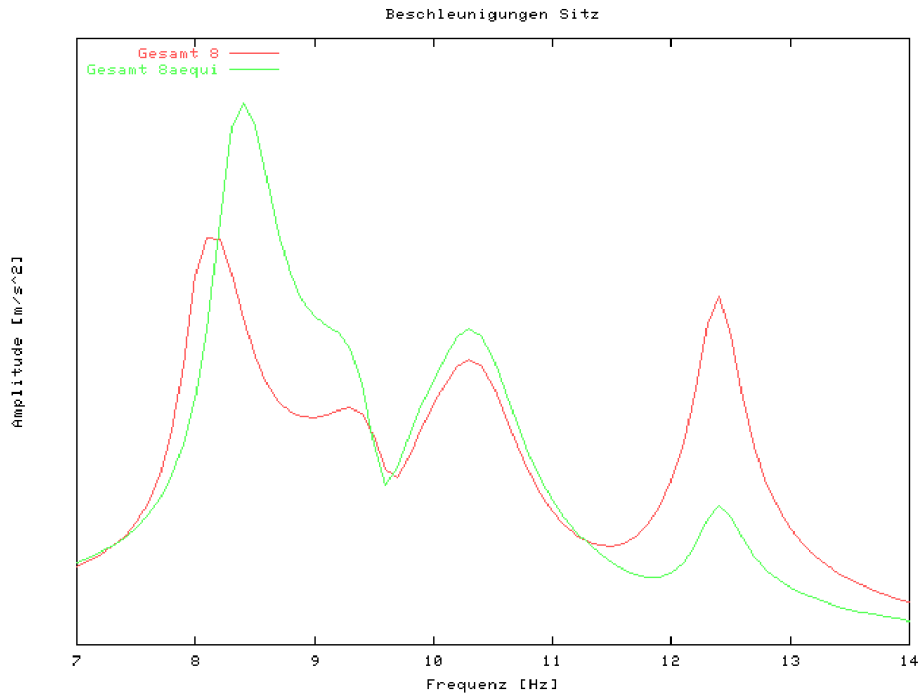

Abbildung 4.29: Geschickte Wahl der Setpoints

Setpointzahl	Rechenzeit	Anzahl SQP-Iterationen	Anteil des QP-Lösens an Rechenzeit
2	4:08	20	5.6%
4	10:52	23	12.0%
4	8:49	19	12.6%
4	5:42	11	12.7%
8	35:54	31	27.5%
8	29:20	28	29.6%
8	33:21	28	27.7%
15	49:33	15	55.7%
16	1:03:28	17	58.0%
16	1:25:45	23	56.9%
23	1:21:35	9	74.3%
30	11:44:19	47	81.0%
32	23:58:15	80	82.4%

Abbildung 4.30 zeigt den mehr als linear anwachsenden Rechenaufwand pro SQP-Iteration bei zunehmender Setpointzahl.

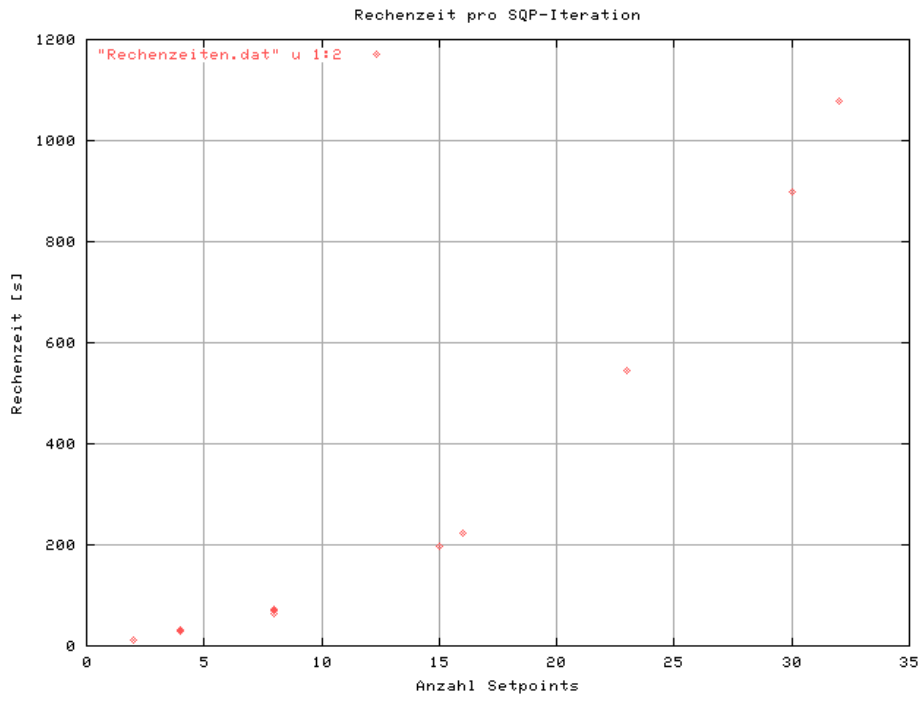

Abbildung 4.30: Rechenzeiten pro SQP-Iteration

5 Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurde eine Klasse von Multiple-Setpoint-Problemen vorgestellt. Es wurde ausgeführt, warum diese Problemformulierung in vielen Anwendungen von großem Nutzen ist und zu praxisgerechten Lösungen führt, die Zulässigkeit und Optimierung einer Zielfunktion für unterschiedliche Szenarien bzw. für ganze Bereiche von Parametern garantieren.
Es wurden die Strukturen dieser besonderen Problemformulierung eingehend untersucht, und aufgrund dessen wurde ein neuer, strukturausnutzender und effizienter SQP-Algorithmus für Multiple-Setpoint-Probleme präsentiert, der es möglich macht, neue Problemklassen zu behandeln bzw. Lösungen zu produzieren, die für den praktischen Einsatz in vielen Fällen geeigneter sind als bei bisherigen Optimierungsansätzen.
Zur Entwicklung dieses Algorithmus wurde ein Überblick über Theorie und Methoden aus dem Bereich der optimalen Steuerung von dynamischen Systemen gegeben und begründet, warum mit Bocks Mehrzielmethode ein geeignetes Diskretisierungsverfahren ausgewählt wurde. Zur Lösung des entstehenden nichtlinearen Problems wurde ein spezielles, die Struktur dieses NLPs ausnutzendes SQP-Verfahren gewählt, in dem die neuesten Techniken aus dem Bereich der nichtlinearen Optimierung eingearbeitet sind.

Insgesamt wurden in dem neuen Algorithmus folgende Techniken verwendet:

- Bocks Mehrzielmethode zur Diskretisierung des kontinuerlichen Problems (Abschnitt 1.3)
- Lösung der Anfangswertprobleme mit einem Adams-Bashforth-Moulton-Verfahren (Abschnitt 1.1.3)
- SQP-Verfahren zur Lösung des entstehenden NLPs mit folgenden Eigenschaften
  - Verwendung eines partiell reduzierten SQP-Verfahrens zur Beschleunigung der Rechenzeiten (Abschnitt 2.2.5)
  - Interne numerische Differentiation zur effizienten und präzisen Berechnung der Ableitungen (Abschnitt 2.3.1)
  - Initialisierung der Hessematrix nach Plitt (Abschnitt 2.2.1.1)
  - Block-Update-Formeln für das Update der Hessematrix (Abschnitt 2.3.3)
  - Update der Hessematrixblöcke mit einer BFGS-Update-Formel mit Powell-Modifikation, um positive Definitheit zu erhalten (Abschnitt 2.2.1.2)
  - Verwendung von Limited-Memory-Update-Formeln, um ein Anwachsen der Kondition der Approximation der Hessematrix zu vermeiden (Abschnitt 2.2.1.2)

  - Kondensierungsalgorithmus zur Reduzierung des quadratischen Programms (Abschnitt 2.3.4)
  - Lösung des quadratischen Programms durch eine Active-Set-Strategie (Abschnitt 2.2.3.3)
  - Liniensuche basierend auf Powells Watchdog-Technik, um globale Konvergenz zu garantieren (Abschnitt 2.2.4)

- Natürliche Koordinaten zur Modellierung des Mehrkörpersystems (Abschnitt 4.2.2)
- Index-Reduktion zur Formulierung der quasi-linearen DAE vom Index 1 (Abschnitt 4.2.3)

Aber es genügt nicht, allein numerische Ansätze zu entwickeln, um problemangepaßte Methoden zu erhalten, mit denen man praxisgerechte Lösungen für komplexe Probleme findet. Von gleicher Wichtigkeit ist es, die Algorithmen in effizienter und flexibler Software umzusetzen, die komfortabel anzuwenden und zu erweitern ist. Mit dem Mehrkörpersimulationstool MBSNAT und der Optimierungsoftware MUSCOD-II als Basis wurde eine Software entwickelt, die diesen Kriterien genügt. Bei der Erstellung dieser Software mußten unter anderem folgende softwaretechnische Arbeiten erledigt werden:

- Erweiterung des Softwarepakets MBSNAT zur Modellierung der äußeren Kräfte (Anregung von Rad und Motor)
- Erweiterung des Softwarepakets MBSNAT zur Berechnung von externer Dynamik (zur Berechnung der Fouriertransformation)
- Erweiterung des Integrators MBSSIM zur Berechnung von Ableitungen
- Anbindung von MBSNAT und MBSSIM an die Optimierungssoftware MUSCOD-II
- Implementierung des neuen SQP-Algorithmus für Multiple-Setpoint-Probleme
- Implementierung von Routinen zur Ansteuerung des Modells für verschiedene Stufen bzw. Setpoints
- Implementierung von geeigneten Eingabefiles zur Ansteuerung der verschiedenen Anregungen
- Implementierung von Ausgaberoutinen zur Erstellung von graphischen Plots

Die Effizienz der Software wurde an einem komplexen Problem aus der Industrie demonstriert, das bisher noch nicht gelöst werden konnte. Das Problem der Schwingungsübertragung in Fahrzeugen wurde mit zwei Ansätzen gerechnet. Der herkömmliche Ansatz mit Einschwing- und Meßphase führt schnell zu sehr großen Rechenzeiten und bringt außerdem einige numerische Schwierigkeiten mit sich. Mithilfe eines neuen Ansatzes auf Basis eines Randwertproblems konnte die Rechenzeit deutlich verkürzt werden, zum Beispiel konnte für eine Anzahl von 16

Setpoints die Rechenzeit um 90% verringert werden. Die erhaltenen Lösungen minimieren die Schwingungsübertragung bei Motoranregung im vorgegebenem Frequenzbereich um ca. 80% gegenüber der gegebenen Startlösung.

Die erfolgreiche Kooperation mit Freudenberg soll fortgesetzt werden. Es gibt Anfragen zu weiteren Optimierungsproblemen aus dem Bereich der Mehrkörpermodelle, zum Beispiel die Optimierung des Fahrkomforts eines Traktors beim Überfahren einer Teststrecke, die aus Balkenschwellen verschiedener Breite und Höhe besteht. Da die übertragene Bewegung auf den Fahrersitz aufgrund der konkreten Teststrecke nicht periodisch ist und von einer Fourieranalyse nicht erfaßt werden kann, ist hier die Formulierung eines anderen Zielfunktionals nötig. Weil die Teststrecke aber mit verschiedenen Geschwindigkeiten überfahren werden soll, ergibt sich auch hier ein Multiple-Setpoint-Problem.

Das Konzept der Multiple-Setpoint-Optimierung läßt sich an die verschiedensten Algorithmen koppeln. So könnte zum Beispiel weitere Rechenzeit eingespart werden durch den Gebrauch einer parallelen Variante der Mehrzielmethode auf einem Parallelrechner. Aber auch für gänzlich andere Verfahren (z.B. Kollokation, Verfahren für PDEs) läßt sich das Konzept anwenden.

Literaturverzeichnis

[Alb06] Jan Albersmeyer. Effiziente Ableitungserzeugung in einem adaptiven BDF-Verfahren. Master's thesis, University of Heidelberg, 2006.

[Asc98] U.M. Ascher. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM, 1998.

[Bau99] I. Bauer. *Numerische Verfahren zur Lösung von Anfangswertaufgaben und zur Generierung von ersten und zweiten Ableitungen mit Anwendungen bei Optimierungsaufgaben in Chemie und Verfahrenstechnik*. PhD thesis, University of Heidelberg, 1999. Download at: http://www.ub.uni-heidelberg.de/archiv/1513.

[BBB+01] T. Binder, L. Blank, H.G. Bock, R. Bulirsch, W. Dahmen, M. Diehl, T. Kronseder, W. Marquardt, J.P. Schlöder, and O.v. Stryk. Introduction to model based optimization of chemical processes on moving horizons. In M. Grötschel, S.O. Krumke, and J. Rambau, editors, *Online Optimization of Large Scale Systems: State of the Art*, pages 295–340. Springer, 2001. http://www.zib.de/dfg-echtzeit/Publikationen/Preprints/Preprint-01-15.html.

[BBLS99] H.G. Bock, I. Bauer, D.B. Leineweber, and J.P. Schlöder. Direct multiple shooting methods for control and optimization of DAE in chemical engineering. In F. Keil, W. Mackens, H. Voß, and J. Werther, editors, *Scientific Computing in Chemical Engineering II*, volume 2, pages 2–18, Berlin, 1999. Springer.

[BC85] L.T. Biegler and J.E. Cuthrell. Improved infeasible path optimization for sequential modular simulators-II: the optimization algorithm. *Computers and Chemical Engineering*, 9:257–267, 1985.

[BCP96] K.E. Brenan, S.L. Campbell, and L.R. Petzold. *Numerical solution of initial-value problems in differential-algebraic equations*. Classics in Applied Mathematics. 14. Philadelphia, PA: SIAM, Society for Industrial and Applied Mathematics. x, 1996.

[BDLS00] H.G. Bock, M. Diehl, D.B. Leineweber, and J.P. Schlöder. A direct multiple shooting method for real-time optimization of nonlinear DAE processes. In F. Allgöwer and A. Zheng, editors, *Nonlinear Predictive Control*, volume 26 of *Progress in Systems Theory*, pages 246–267, Basel, 2000. Birkhäuser.

[BEKS02] H.G. Bock, W. Egartner, W. Kappis, and V. Schulz. Practical shape optimization for turbine and compressor blades by the use of PRSQP methods. *Optimization and Engineering*, 3(4):395–414, 2002.

[Bie84] L.T. Biegler. Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation. *Computers and Chemical Engineering*, 8:243–248, 1984.

[BKS00] H.G. Bock, E.A. Kostina, and J.P. Schlöder. On the role of natural level functions to achieve global convergence for damped Newton methods. In M. Powell et al., editor, *System Modelling and Optimization. Methods, Theory and Applications*. Kluwer, 2000.

[BNS95] L.T. Biegler, J. Nocedal, and C. Schmid. A reduced Hessian method for large-scale constrained optimization. *SIAM Journal on Optimization*, 5:314–347, 1995.

[Boc81] H.G. Bock. Numerical treatment of inverse problems in chemical reaction kinetics. In K.H. Ebert, P. Deuflhard, and W. Jäger, editors, *Modelling of Chemical Reaction Systems*, volume 18 of *Springer Series in Chemical Physics*, pages 102–125. Springer, Heidelberg, 1981.

[Boc83] H.G. Bock. Recent advances in parameter identification techniques for ODE. In P. Deuflhard and E. Hairer, editors, *Numerical Treatment of Inverse Problems in Differential and Integral Equations*. Birkhäuser, Boston, 1983.

[Boc87] H.G. Bock. *Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen*, volume 183 of *Bonner Mathematische Schriften*. University of Bonn, Bonn, 1987.

[BP84] H.G. Bock and K.J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Proc. 9th IFAC World Congress Budapest*, pages 243–247. Pergamon Press, 1984.

[BP03] U. Brandt-Pollmann. Nicht-Standard Probleme der optimalen Steuerung in Chemie und Biotechnologie. Technical report, DECHEMA Jahrestagung der Biotechnologen, München, April 2003.

[BP04] U. Brandt-Pollmann. *Numerical solution of optimal control problems with implicitly defined discontinuities with applications in engineering*. PhD thesis, IWR, University of Heidelberg, 2004.

[BPDLP04] U. Brandt-Pollmann, M. Diehl, D. Lebiedz, and A. Potschka. A parallel optimal control algorithm based on direct multiple shooting for differential algebraic equations. *in Preparation*, 2004.

[BT96] D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

[BTN02] A. Ben-Tal and A. Nemirovski. Robust optimization - methodology and applications. *Mathematical Programming*, 92, 2002.

[CLPP82] R. Chamberlain, C. Lemaréchal, H. C. Pedersen, and M. J. D. Powell. The watchdog technique for forcing convergence in algorithms for constrained optimization. *Mathematical Programming*, 16:1–17, 1982.

[Cry72] C.W. Cryer. On the instability of high order backward-difference multistep methods. *BIT*, 12:17–25, 1972.

[DBK06] M. Diehl, H.G. Bock, and E. Kostina. An approximation technique for robust nonlinear optimization. *Math. Prog. Series B*, 2006. (in print).

[DFS+02] M. Diehl, R. Findeisen, S. Schwarzkopf, I. Uslu, F. Allgöwer, H.G. Bock, E.D. Gilles, and J.P. Schlöder. An efficient algorithm for nonlinear model predictive control of large-scale systems. Part I: Description of the method. *Automatisierungstechnik*, 50(12):557–567, 2002.

[Die98] M. Diehl. A direct multiple shooting method for the optimization and control of chemical processes. Master's thesis, University of Heidelberg, 1998.

[Die02] M. Diehl. *Real-Time Optimization for Large Scale Nonlinear Processes*, volume 920 of *Fortschr.-Ber. VDI Reihe 8, Meß-, Steuerungs- und Regelungstechnik*. VDI Verlag, Düsseldorf, 2002. Download also at: http://www.ub.uni-heidelberg.de/archiv/1659/.

[Die04] M. Diehl. An approximation technique for robust nonlinear optimization. *Mathematical Programming*, 2004. (submitted).

[Diw95] U.M. Diwekar. *Batch distillation: simulation, optimal design, and control*. Taylor & Francis, Washington, 1995.

[DS83] J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Computational Mathematics. Prentice Hall, 1983.

[Ega99] W. Egartner. Working range optimization for turbine and compressor blading. *J. Comput. Appl. Math.*, 120(1-2):59–65, 1999.

[Eic91] E. Eich. *Projizierende Mehrschrittverfahren zur numerischen Lösung von Bewegungsgleichungen technischer Mehrkörpersysteme mit Zwangsbedingungen und Unstetigkeiten*. PhD thesis, University of Augsburg, 1991.

[FL02] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. *Mathematical Programming*, 91(2):239–269, 2002.

[Fle82] R. Fletcher. Second order corrections for nondifferentiable optimization. *Numerical analysis (Dundee, 1981)*, 912:85–114, 1982.

[Fle87] R. Fletcher. *Practical Methods of Optimization*. Wiley, Chichester, 2 edition, 1987.

[FMT02] R. Franke, M. Meyer, and P. Terwiesch. Optimal control of the driving of trains. *Automatisierungstechnik*, 50(12), 2002.

[Gab82] D. Gabay. Reduced quasi-newton methods with feasibility improvement for nonlinear constrained optimization. *Mathematical Programming*, 16:18–44, 1982.

[GB94] J.V. Gallitzendörfer and H.G. Bock. Parallel algorithms for optimization boundary value problems in DAE. In H. Langendörfer, editor, *Praxisorientierte Parallelverarbeitung*. Hanser, München, 1994.

[GdJB94] J. García de Jalón and E. Bayo. *Kinematic and Dynamic Simulation of Multibody Systems*. Springer, Berlin, 1994.

[Gea88] C.W. Gear. Differential-algebraic equation index transformations. *SIAM J. Sci. Stat. Comp.*, 9:39–47, 1988.

[Gri00] A. Griewank. *Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation*. Number 19 in Frontiers in Appl. Math. SIAM, Philadelphia, 2000.

[HW93] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*. Springer Series in Computational Mathematics, 1993.

[KKBS04] S. Körkel, E. Kostina, H.G. Bock, and J.P. Schlöder. Numerical methods for optimal control problems in design of robust optimal experiments for nonlinear dynamic processes. *Optimization Methods and Software*, 19:327–338, 2004.

[KKEvS01] A. Kröner, T. Kronseder, G. Engl, and O. von Stryk. Dynamic optimization for air separation plants. *Proceedings of the European Symposium on Computational Aided Process engineering (ESCAPE-11)*, 2001.

[Kör02] S. Körkel. *Numerische Methoden für Optimale Versuchsplanungsprobleme bei nichtlinearen DAE-Modellen*. PhD thesis, University of Heidelberg, Heidelberg, 2002.

[Kra97] C. Kraus. Modellierung und rekursive Algorithmen für Mehrkörpersysteme in Natürlichen Koordinaten. Master's thesis, University of Heidelberg, 1997.

[Kra05] C. Kraus. *Efficient Object-Oriented Modelling, Simulation and Parameter Estimation for Biomechanical Problems*. PhD thesis, University of Heidelberg, 2005.

[KvSB01] T. Kronseder, O. von Stryk, and R. Bulirsch. Towards nonlinear model based predictive optimal control of large-scale process models with application to air separation plants. In M. Grötschel, S.O. Krumke, and J. Rambau, editors, *Online Optimization of Large Scale Systems: State of the Art*, pages 385–412. Springer, 2001.

[LBBS03] D.B. Leineweber, I. Bauer, H.G. Bock, and J.P. Schlöder. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part I: Theoretical aspects. *Computers and Chemical Engineering*, 27:157–166, 2003.

[Lei95] D.B. Leineweber. Analyse und Restrukturierung eines Verfahrens zur direkten Lösung von Optimal-Steuerungsproblemen. Master's thesis, University of Heidelberg, 1995.

[Lei99] D.B. Leineweber. *Efficient reduced SQP methods for the optimization of chemical processes described by large sparse DAE models*, volume 613 of *Fortschr.-Ber. VDI Reihe 3, Verfahrenstechnik*. VDI Verlag, Düsseldorf, 1999.

[LSBS03] D.B. Leineweber, A.A.S. Schäfer, H.G. Bock, and J.P. Schlöder. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part II: Software aspects and applications. *Computers and Chemical Engineering*, 27:167–174, 2003.

[Mar78] N. Maratos. *Exact penaly function algorithms for finite-dimensional and control optimization problems*. PhD thesis, Imperial College, London, 1978.

[Mar02] W. Marquardt. Nonlinear model reduction for optimization and control of transient chemical processes. In J.B. Rawlings, B.A. Ogunnaike, and J.W. Eaton, editors, *Process Control VI. AIChE Symp.*, volume 98 of *326*, pages 12–42, 2002.

[Mey04] Dany Meyer. *Modellbasierte Mehrzieloptimierung mit Neuronalen Netzen und Evolutionsstrategien*. PhD thesis, Technische Universität Ilmenau, 2004.

[NO85] J. Nocedal and M.L. Overton. Projected Hessian updating algorithms for nonlinear constrained optimization. *SIAM Journal on Numeric al Analysis*, 22:821–850, 1985.

[Noc80] J. Nocedal. Updating quasi-Newton matrices with limited storage. *Math. of Comput.*, 35:773–782, 1980.

[NW99] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer, 1999.

[PA04] I. Papamichail and C.S. Adjiman. Global optimization of dynamic systems. *Computers and Chemical Engineering*, 28:403–415, 2004.

[PCC97] C. Pierre, M.P. Castanier, and S. Choi. On developing new statistical energy methods for the analysis of vibration transmission in complex vehicle structures. *Mechanics of Structures and Machines*, 25, No. 1:87–101, 1997.

[Pli81] K.J. Plitt. Ein superlinear konvergentes Mehrzielverfahren zur direkten Berechnung beschränkter optimaler Steuerungen. Master's thesis, University of Bonn, 1981.

[Pot06] A. Potschka. Inequality state constraints in optimal control problems. Master's thesis, University of Heidelberg, 2006.

[Pow78] M.J.D. Powell. Algorithms for nonlinear constraints that use Lagrangian functions. *Mathematical Programming*, 14(3):224–248, 1978.

[Pow85] M.J.D. Powell. On the quadratic programming algorithm of Goldfarb and Idnani. *Mathematical Programming Study*, 25:46–61, 1985.

[RBP+99] R. Ross, V. Bansal, J.D. Perkins, E.N. Pistikopoulos, J.M.G. van Schijndel, and G.L.M. Koot. Optimal design and control of an industrial distillation system. *Computers and Chemical Engineering*, 23(SS):S875–S878, 1999.

[Rie01] P. Riede. Entwicklung eines parallelen PRSQP-Verfahren zur Lösung von Multiple-Setpoint-Optimierungsproblemen. Master's thesis, University of Heidelberg, 2001. Download at: http://www.rzuser.uni-heidelberg.de/ jx7/diplom.ps.

[Rob74] S.M. Robinson. Perturbed Kuhn-Tucker points and rates of convergence for a class of nonlinear programming algorithms. *Mathematical Programming*, 7:1–16, 1974.

[Sag05] S. Sager. *Numerical methods for mixed–integer optimal control problems*. PhD thesis, University of Heidelberg, Tönning, Lübeck, Marburg, 2005.

[SBD+05] S. Sager, H.G. Bock, M. Diehl, G. Reinelt, and J.P. Schlöder. Numerical methods for optimal control with binary control functions applied to a Lotka-Volterra type fishing problem. In A. Seeger, editor, *Recent Advances in Optimization (Proceedings of the 12th French-German-Spanish Conference on Optimization)*, Lectures Notes in Economics and Mathematical Systems. Springer, 2005. (to appear).

[SBS98] V.H. Schulz, H.G. Bock, and M.C. Steinbach. Exploiting invariants in the numerical solution of multipoint boundary value problems for DAEs. *SIAM Journal on Scientific Computing*, 19:440–467, 1998.

[Sch96] V.H. Schulz. *Reduced SQP methods for large-scale optimal control problems in DAE with application to path planning problems for satellite mounted robots*. PhD thesis, University of Heidelberg, 1996.

[Sch04] A.A.S. Schäfer. *Effiziente reduzierte Newton-ähnliche Verfahren zur Behandlung hochdimensionaler strukturierter Optimierungsprobleme mit Anwendung bei biologischen und chemischen Prozessen*. PhD thesis, IWR Universität Heidelberg, http://www.ub.uni-heidelberg.de/archiv/5264, 2004.

[Sch05] Martin Schlegel. *Adaptive discretation methods for the efficient solution of dynamic optimization problems*. RWTH AAchen, 2005.

[Sha86] L.F. Shampine. Conservation laws and the numerical solution of ODEs. *Computational and Mathematics with Applications*, 12B:1287–1296, 1986.

[SR03] A. Shapiro and A. Ruszczynski. *Stochastic Programming*. Number 0444508546 in Handbooks in Operations Research and Management Science. Elsevier Publishing Company, 2003.

[SS78] R.W.H. Sargent and G.R. Sullivan. The development of an efficient optimal control package. In J. Stoer, editor, *Proceedings of the 8th IFIP Conference on Optimization Techniques (1977), Part 2*, Heidelberg, 1978. Springer.

[Str93] O. von Stryk. Numerical solution of optimal control problems by direct collocation. In *Optimal Control: Calculus of Variations, Optimal Control Theory and Numerical Methods*, volume 129. Bulirsch et al., 1993.

[TBK04] S. Terwen, M. Back, and V. Krebs. Predictive powertrain control for heavy duty trucks. In *Proceedings of IFAC Symposium in Advances in Automotive Control*, Salerno, Italy, 2004.

[VG05] VDI-Gesellschaft. *Einwirkung mechanischer Schwingungen auf den Menschen - Ganzkörperschwingungen an Arbeitsplätzen in Gebäuden*. VDI-Gesellschaft Entwicklung Konstruktion Vertrieb, 2005.

[vS99] R. von Schwerin. *MultiBody System SIMulation: Numerical Methods, Algorithms, and Software*. Springer, 1999.

[VSP94] V.S. Vassiliadis, R.W.H. Sargent, and C.C. Pantelides. Solution of a class of multistage dynamic optimization problems. 1. problems without path constraints. *Industrial and Engineering Chemistry Research*, 10(33):2111–2122, 1994.

[vSW94] R. von Schwerin and Michael Winckler. *A Guide to the Integrator Library MBS-SIM - Version 1.00*. IWR-Preprint 94-75, University of Heidelberg, 1994.

[WB02] A. Wächter and A.T. Biegler. Global and local convergence of line search filter methods for nonlinear programming. Technical report, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, 2002.

[WBPMS04] R. Winkler, U. Brandt-Pollmann, U. Moslener, and J.P. Schlöder. Time-lags in capital accumulation. In *Operations Research Proceedings*. D. Ahr, R. Fahrion, M. Oswald, and G. Reinelt, 2004.

[Wil63] R.B. Wilson. *A simplicial algorithm for concave programming*. PhD thesis, Harvard University, 1963.

[Wri97] S.J. Wright. *Primal-Dual Interior-Point Methods*. SIAM Publications, Philadelphia, 1997.

[Zha05] Yin Zhang. A general robust-optimization formulation for nonlinear programming. Technical Report (Technical Report, TR04-13), Department of Computational and Applied Mathematics, Rice University, Houston, Texas, U.S.A., 2005.